UNIVERSITÉ DE MONTRÉAL

COMPUTATION OF NEUTRON FLUXES IN FUEL PINS ARRANGED IN HEXAGONAL LATTICES

HEM PRABHA RAGHAV
DÉPARTEMENT DE GÉNIE PHYSIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉNERGÉTIQUE)
FÉVRIER 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

COMPUTATION OF NEUTRON FLUXES IN FUEL PINS ARRANGED IN HEXAGONAL LATTICES

présenté par: RAGHAV Hem Prabha
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. KOCLAS Jean, Ph.D., président.
M. MARLEAU Guy, Ph.D., membre et directeur de recherche.
M. ÉTIENNE Stéphane, Doct., membre.

ACKNOWLEDGMENTS

I am thankful to Prof. Alain Hébert and Prof. Guy Marleau for giving me this opportunity as well as for their guidance. I am also thankful to Prof. Jean Koclas and Prof. Andrei for their guidance. This was the continuous support and guidance of Prof. Guy Marleau that this work could be completed. I am grateful to him for his infinite patience and tolerance towards my limited programing and writing skills. Inspite of his busy schedule he always spared time to guide.

Thanks are due to Prof. Jean Koclas and Prof. Stéphane Étienne for accepting to be in the jury.

Like some people, it was my long standing desire to do research in west. Due to one reason or other I did not make use of the opportunities earlier. Dr. Karthikeyan had all the praise for the institute and the people around here. Over a period of time I seem to agree with him. I am thankful to Karthikeyan, Alexi, Marc, Nicolas, Remi, Thibaud, Hadrien, Mathieu, Majid, Altan, Tarek and other students, they were always there to help in need. I am also thankful to Jolly, Lakshmi, Ramaa, Sujata and Mary who were there to help me in adjusting to a different enviornment. Finally I am thankful to my family and friends who gave me constant encouragement and support.

RÉSUMÉ

Pour comprendre le comportement d'un réacteur nucléaire, une description détaillée de la population de neutrons (flux de neutrons) dans le coeur est requise. Le coeur du réacteur nucléaire peut être représenté par un réseau répétitif de cellules unitaires. Chaque cellule de ce réseau se compose soit d'un crayon de combustible unique ou d'un groupe de crayons entourés par le modérateur. Ces cellules peuvent être disposées pour former un réseau cartésien ou hexagonal, en fonction du type de réacteur nucléaire. Comme la majorité des réacteurs actuels (réacteurs à eau pressurisée, bouillante et CANDU) possèdent un réseau ayant un pas carré, le module NXT de DRAGON qui permet d'analyser de tels réseaux de cellules s'est avéré jusqu'à présent adéquat. Cependant, pour les réacteurs avancés qui sont déjà sur la table de dessin, les cellules sont plutôt disposées sur un réseau hexagonal. La possibilité de procéder à de telles analyses en utilisant DRAGON est cependant assez limitée, car le module NXT ne permet pas pour le moment d'analyser des réseaux hexagonaux alors que le type de géométries hexagonales pouvant être traitées par le module EXCELT est très limité et mal adapté aux nouveaux réacteurs. Ici, nous proposons de généraliser le module NXT afin de permettre l'analyse de réseaux de cellules hexagonales.

Nous avons utilisé la méthode des probabilités de collision de Carlvik pour effectuer les calculs de transport requis pour déterminer la distribution spatiale de flux dans un assemblage. Cette méthode permet de prendre en compte les détails géométriques exacts d'une cellule réacteur en deux et trois dimensions. Elle requiert un algorithme de quadrature numérique des probabilités de collision. Cet algorithme est basé sur une méthode de traçage de lignes d'intégration et il peut être appliqué facilement à un assemblage de cellules hexagonales.

Afin d'atteindre nos objectifs, nous avons premièrement développé, en FORTRAN, notre algorithme de traçage de lignes d'intégration pour des géométries hexagonales en deux dimensions. Après l'évaluation de la longueur de chaque segment de droite croisant une région spécifique de la cellule en deux dimensions, nous avons développé une technique qui permet de reconstruire ces segments de droite dans la troisième dimension. Tout ce travail a été programmé dans le module NXT du code DRAGON. Le choix de ce module réside dans sa capacité de combiner automatiquement aux segments de droites provenant de géométries hexagonales ou cartésiennes ceux associées à des géométries cylindriques ou de grappes en deux et trois dimensions (déjà disponible dans NXT), à la condition que ces cylindres et grappes soient complètement inclus dans une cellule.

Pour valider notre méthode de traçage de lignes d'intégration, les réseaux suivants ont été analysés :

a) cellules hexagonales contenant des sous-régions cylindriques en 2D et 3D.

b) réseau de sept cellules hexagonales chacune contenant des sous-régions cylindriques en 2D et 3D.

c) cellule hexagonale contenant des sous-régions cylindriques en plus de grappes de crayons de combustibles 2D et 3D.

Les résultats de ces analyses sont ensuite comparés, lorsque possibles, à ceux obtenus en utilisant le module EXCELT de DRAGON. Les lignes d'intégration sont également vérifiées indépendamment en utilisant l'option de DRAGON permettant de préparer un fichier qui peut être utilisé par Matlab pour illustrer ces lignes.

Cet exercice a démontré que le module NXT peut analyser des réseaux hexagonaux de cellules contenant des sous-régions annulaires et des grappes de combustible.

ABSTRACT

To understand the behavior of a nuclear reactor, a detailed description of the neutron population or flux, in the reactor core, is required. A nuclear reactor core consists of repetitive arrangement of many unit lattice cells. A lattice cell consists of either a single fuel pin or a cluster of many fuel pins surrounded by moderator. These lattice cells are arranged in a square or hexagonal lattice pitch, depending of the type of nuclear reactor. Most of the present pressurized water reactors have square pitch. Advanced reactors have hexagonal pitch. The NXT module of the code DRAGON, which is developed at Ecole Poytechnique, computes neutron flux distribution in a reactor geometry, when the lattice cells are arranged in a square pitch. Using the contributions of this thesis, computations are extended to a reactor geometry when the lattices cells are arranged in a hexagonal pitch.

We have considered Carlvik's method of collision probabilities for computations. This method can consider exact geometrical details of a reactor lattice cell in two and three dimensions. While using this method, tracking algorithms are required to be developed. Initially we developed, in FORTRAN, tracking algorithm for two dimensional hexagonal geometries. This algorithm is based on the ray tracing method and it can be applied to assembly of hexagonal cells. After computing the track lengths in two dimensions, we have developed equations and have applied these equations to compute tracks in the third dimension. All this work has been implemented in the NXT module of the code DRAGON. Advantage of the NXT module is that, it can compute tracks in cylindrical fuel pins in a two dimensional (2D) or three dimensional (3D) lattice cell, provided tracks of the outer boundary of the cell are given. Earlier NXT module could consider square and cube geometries, now with the implementation of this algorithm it can consider hexagonal assemblies in 2D and 3D.

For testing the results so obtained, the following lattices have been analyzed and results are compared with the EXCELT module of the code DRAGON. All the results have also been verified by plotting them (NXT module does this plotting by using Matlab). Results are computed, in one group, for the following lattice cells

a) Single hexagonal pin cells in 2D and 3D.

b) Assembly of seven hexagonal single cylindrical fuel pin cells in 2D and 3D.

c) Single hexagonal cell with a cluster of pins in 2D and 3D. This exercise was done to demonstrate that the NXT module can compute fluxes in hexagonal cell with a cluster of pins in 2D and 3D. For doing this, care has to be taken such that co-ordinate systems, for track length computations, in hexagon and in pins should match. The EXCELT module of DRAGON code can compute fluxes in a single hexagonal cell with a cluster of pin cells in 2D and not in 3D. All this work has been presented in this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In order to completely understand the behavior of a nuclear reactor, a detailed description of its neutron population or flux is required. A general strategy developed for neutronic computations for the entire reactor core includes three steps: (a) unit pin cell level, (b) full assembly level (consisting of many unit cells along with control rods, structural material etc) and full core analysis in three dimensional space. The unit cell of a reactor may have different geometry depending on the reactor type. Typical unit cell of a nuclear reactor consists of a single cylindrical fuel pin or a cluster of many such pins surrounded by moderator. These pins can be arranged in a square or a hexagonal pitch. Numerical methods based on transport equation are developed at the unit cell level and at assembly level. As the computation power increases more complex geometries can be considered.

Detailed computations done at unit cell level are used to compute average cross-sections for the unit cell. These average cross-sections for the cell are used, in less time consuming methods i.e. diffusion theory for the entire reactor geometry. Lattice cell computations at the Nuclear Engineering Institute are done by the computer code DRAGON [24, 25]. The reactor computations are done by the code DONJON [36, 39]. These codes are continuously developed. With new reactor designs, new computational methods are required to be implemented in these codes.

There has been a continuous effort to design new advanced nuclear reactors and develop new computational methods to design these reactors. Our effort, in this thesis, is towards development of new computational methods for the design of advanced reactors and their implementation in the DRAGON code. Before describing the work, we will present a review of the literature.

1.2 Review of the literature

Some of the improved design of nuclear reactors are PWRs (Pressurized Water Reactors-AP1000), ACR 1000 [14] and VVER [31] (Russian reactors cooled and moderated by light water). AP1000 is a PWR model by Westinghouse [34]. It uses light water as coolant-moderator and requires fuel enrichment up to 4.95%. ACR (Advanced Candu Reactor) is the next generation of CANDU reactors and combines features of both PHWR (Pressurized

Heavy Water Reactors) [11] and PWR. It was originally named NG-CANDU [8] and employs light water as coolant but heavy water as moderator. CANDU-6 [9] uses natural Uranium as fuel while ACR uses enriched fuel.

The designs of generation IV reactors are different from the designs of present reactors.

1.2.1 Advanced Reactor Designs

Since nuclear reactors can reach high temperatures, generation IV reactors are designed to produce hydrogen. Generation IV gas-cooled fast reactor (GFR) makes use of ceramic plate-type fuel-elements with (U-Pu) carbide fuel contained within a SiC inert matrix. Some of the other designs of these reactors are Pebble Bed Reactors (PBR) [15] and VHTR [18, 38, 10, 28, 19]. The pebble bed reactor (PBR) is a graphite-moderated, gas-cooled, nuclear reactor. It is a type of Very high temperature reactor (VHTR) formally known as the high temperature gas reactor (HTGR). The designs of PBR and VHTR use TRISO fuel, which allow high outlet temperatures and passive safety.

These TRISO fuel particles consist of a fissile material (such as U235) surrounded by a coated ceramic layer of SiC for structural integrity. In these designs about 360,000 pebbles are placed together to create a reactor, and are cooled by an inert or semi-inert gas such as helium, nitrogen or carbon dioxide. Also, the gases do not dissolve contaminants or absorb neutrons as water does, so the core has less in the way of radioactive fluids.

Currently, the HTRs have been receiving significant attention due to many desired characteristics such as inherent safety, modularity, relatively low cost, short construction period, and easy financing. Graphite is the moderator in the core, and can at the same time be utilized as a structure material. Presently for VHTR design, the spheres are smeared to make it a cylindrical geometry. These cylinders are put in a hexagonal lattice. Presence of control rods in these assemblies introduces asymmetry in the hexagonal lattice.

Problems arise as how to consider neutron population or flux in such geometries? These geometries are different from the geometries of present reactors. In this thesis we have considered hexagonal lattice. Hexagonal lattices are also present in other reactor geometries such as VVER etc. Before describing the work we will review some of the available methods of computations.

1.2.2 Methods of computations

The need for accurate modeling in the design of advanced nuclear fuel assemblies has motivated the development of advanced particle transport codes. Historically there have been two standard ways of approaching this problem. First is the stochastic or Monte Carlo ap-

proach [4]. This approach uses random numbers to track the paths of numerous hypothetical neutrons and their probabilistic interaction with nuclei. With a sufficiently large number of particles, this approach is able to statistically predict the behavior of the system. Second option is to consider a deterministic approach that attempts to solve the transport equation using numerical methods.

1.2.3 Monte Carlo Approach

The main advantage of this approach is the ability to exactly describe the problem both in terms of geometry and physics. This approach is not limited to any specific geometry configuration, and the continuous energy cross-sections allow the physics of the particle interactions to be modeled exactly. This approach has the disadvantage of statistical errors. To increase the accuracy, one has to follow more particles in the problem domain. The disadvantage is the time it takes to run enough particles so that the error is sufficiently small. This is true for problems with large domains, small zones of interest, or high particle losses. Problems with large domain require a large number of particles to cover the domain. Problems with small zones of interest are difficult because one is concerned with the statistical accuracy in the zone. In order to increase the solution accuracy, a large number of particles must pass through the domain. However if the zone is small compared to the entire problem domain, getting a sufficient number of particles in the zone can be difficult. This problem makes one pursuing alternative deterministic methods worthwhile.

1.2.4 Deterministic Approach

The deterministic approach can be broken down in two main categories based on the form of the transport equations considered: the integral form of the transport equation and the integro-differential form of the neutron transport equation. In its integral form, the transport equation is solved for the angle independent (scalar) flux. One of the methods of solving this equation is the collision probability (CP) method (Pij method, Interface current method) [7, 21, 37].

In the collision probability method isotropic scattering is assumed, the integrated fluxes are solved. The spatial coupling is achieved through collision probabilities calculated from one region to all other regions. In the interface current method also isotropic scattering is assumed, fluxes (or currents) at the interfaces are expanded in terms of cosines and sines of the angles subtended by the neutron direction with the interface normal (or spherical harmonics). The treatment of the angular distribution of neutrons consists of coupling these currents in a systematic manner. Spatially one region is connected only to its neighboring

regions through interface currents. This means that coupling coefficient of different angular modes (or transmission probabilities) need to be calculated only for each geometrically and materially unique region.

There are many methods which can be used to solve the integro-differential form of the equation. They mainly differ in the type of discretization used. Some of the commonly used methods are Sn method [5, 26] and Method of characteristics (MOC) [1, 17].

The Sn method is the result of a discrete ordinates discretization of the angle variables. In the method of characteristics, the spatial zone in discretized into a number of zones, within which sources are assumed to be constant or flat. The MOC in turn can perform zone based integration of the transport equation along the characteristic rays.

1.2.5 Computer Codes

Some of the work for hexagonal geometries has been done by using the complete collision probability method for arbitrary combinatorial geometries HELIOS (2D) [40, 12] and GTRAN2 [41]. Some of the proprietary code like WIMS8 [42] can do such computations. The W-THES [35] module calculates approximate cylindrical collision probabilities using the method of Bonalumi [3]. The W-PIP module calculates the flux solution. The W-CACTUS5 solves the multi-group transport equations in (x,y) geometry using the "method of characteristics", a numerical solution to the differential Boltzmann equation. The user specifies the maximum allowable separation between the 2D track mesh. Both azimuthal-and polar-angular integrations are performed numerically, with user-specified angular mesh for the transport approximation. The DRAGON code can handle some of the hexagonal lattices.

1.3 DRAGON code and its features

The DRAGON code is a lattice code which is used to solve the neutron transport equation for a nuclear reactor geometry. This code is under continuous development at Ecole Polytechnique of Montreal since 1986. It is divided into many calculation modules which are linked together by using the GAN generalized driver [32]. Here the exchange of information is ensured by well defined data structures. This is done so that new calculation techniques can be implemented in this code. Here the tracking data for various reactor geometries contains details of numbering of volumes, volume and surface connections and neutron trajectories. This data file enables one to use various methods i.e. the method of collision probabilities (P_{ij} method, Interface current method), the method of characteristics or the Monte Carlo method.

There are various ray tracings methods available in the code DRAGON. Limited tracking

for 1D and 2D, by using collision probability method, is performed by SYBILT module [13]. For some 2D and 3D geometries, the tracking can be performed by EXCELT [30]. This module can compute tracks in isolated clusters including hexagonal geometry [29, 30]. Tracking can also be done in 2D and 3D by the NXT module [23], for geometries in a square pitch. NXT module is a generalization of the EXCELT module. However the NXT module of DRAGON did not have the capability to *track* hexagonal lattices with sufficient detail, considering the requirements of modern computational schemes.

1.3.1 Description of the code DRAGON:

It consists of many modules, which can be modified independently. It can be installed on work stations that support a FORTRAN compiler. Some of the modules of this code have been used in this thesis. These are

- GEO: module that is used to generate and modify a geometry.
- EXCELT: tracking module for 2D and 3D geometries in both rectangular and hexagonal geometries using the collision probability P_{ij} method. It can also track isolated 2D clusters.
- NXT: tracking module for two-dimensional or three-dimensional mixed rectangular and cylindrical geometries using the collision probability P_{ij} method. We have implemented our tracking algorithms in this module.
- TLM: module used to generate a Matlab M-file to obtain a graphics representation of the NXT: tracking lines.
- ASM: module, to prepare the group-dependent complete collision probability or the assembly matrices.
- FLU: module, to solve the multigroup neutron transport equation using the collision probability method.
- EDI: module, which supplies the main editing options where an equivalence method based on the SPH procedure is available.

1.4 Objectives of the research

The `NXT` [23] module of the code DRAGON analyzes geometry, when these pins are arranged in a square pitch. The main objective of the research is to open the NXT module to represent fuel pins arranged in hexagonal lattices. Cells arranged in an hexagonal lattice are

tight packed lattices [16, 27]. In this research, we have considered the method of collision probabilities.

Due to the complexity of designing a general tracking algorithm, programming work is performed using a cycle development model. Each development cycle involves the following steps

1. Analysing the geometry
2. Volume and surface numbering (integer numbers)
3. Compute the volumes and surfaces (real numbers)
4. Generate the tracking
5. Plot the tracking for verification purposes.

We will describe in brief, the methodology used in the thesis.

1.5 Methodology

While using the method of collision probabilities, we encounter two problems

1. Numbering of the regions and surfaces.
2. Computation of the required tracks.

1.5.1 Numbering of the regions and surfaces

Numbering of regions and surfaces is mentioned in chapter 2.

1.5.2 Computation of the required tracks

Initially tracks are computed for one hexagon in 2D. Then these are used to compute tracks in 3D and then in seven hexagonal assemblies (in 2D and 3D).

1.5.3 Computation of tracks in 2D

Here intersections points of a given line with the lines passing through the vertices of the hexagons are considered. By using these intersection points, 2D tracks in the hexagons are computed. These are discussed in chapter 3 (for one hexagon cell) and in Chapter 5 (for assembly hexagon cells in 2D and 3D).

1.5.4 Computation of tracks in 3D

After computing the tracks in 2D, equations are developed to compute tracks in 3D. The tracking algorithms are discussed in chapter 4 (one hexagon) and in chapter 5 (for assembly of 3D hexagonal cells).

Contribution of the thesis is that algorithms are developed to compute tracks in hexagonal geometry. Tracks inside pins are computed by other routines, already programmed in the NXT module. However care has to be taken such that the co-ordinate systems match while combining both sets of tracks. These tracks are used to compute collision probabilities. The expressions for collision probabilities require computation of integrals.

1.5.5 Computation of Integrals

Collision probabilities are expressed in terms of double (2D geometries) or four integrals (3D geometries) (depending on the geometry under consideration). These integrals are evaluated numerically in the ASM: module, we will not discuss these here. Equal weights quadrature methods [6] have been used to compute these integrals, in this work.

1.6 Contents of the thesis

We have developed algorithms, in 2D and 3D, for two types of geometries: single hexagonal lattice cell and assemblies of hexagonal cells. The program for computing tracks has been written in FORTRAN and is implemented in the NXT module of DRAGON (Version 3.06) [24]. With the work of this thesis, NXT module can handle fuel pins arranged in an hexagonal lattice.

It is presented here with increased levels of complexity. Each step terminates with the generation of a complete set of tracking information, in a format compatible with the DRAGON code .

The flux distributions are obtained for corresponding unitary tests. The geometries of increased complexity are considered, these are described below.

1. Single hexagonal pin cell in 2D

   This type of geometry is depicted in Fig. 1.1. Similar tracking algorithms already exist in both SYBILT and EXCELT modules of DRAGON. This first step consists mainly to test the present algorithm and implement it in NXT.

2. Single hexagonal cluster in 2D

This type of geometry is depicted in Fig. 1.2. A similar tracking algorithm already exists in EXCELT module and in the NXT module of DRAGON for Cartesian mesh. This first step consists mainly to test the present algorithm and implement it in NXT for hexagonal geometry.

3. Single hexagonal pin cell in 3D

   This type of geometry is depicted in Fig. 1.3. Similar tracking algorithm already exists in the EXCELT modules of DRAGON. This first step consists mainly to test the present algorithm and implement it in NXT.

4. Single hexagonal cluster in 3D

   This type of geometry is depicted in Fig. 1.4. Such a tracking algorithm does not exist in DRAGON. A similar tracking algorithm already exists in the NXT module of DRAGON for Cartesian mesh. This is generalized to hexagonal geometry.

5. Seven hexagonal pin cells in 2D

   This type of geometry is depicted in Fig. 1.5. A similar tracking algorithm already exists in the EXCELT module of DRAGON. This step consists mainly to test the present algorithm and implement it in NXT.

6. Seven hexagonal pin cells in 3D

   This type of geometry is depicted in Fig. 1.6. This step consists mainly to test the present algorithm and implement it in NXT.

In Chapter 2 the collision probability method and the flux computations, in general, are discussed. Chapter 3 presents algorithms developed for computing tracks, in 2D, for one hexagon cell. The results obtained by these algorithms are compared, when possible, with the results of EXCELT module of DRAGON. Chapter 4 presents algorithm developed for computing tracks for one finite height hexagon cell (3D). The results obtained by these algorithms, in 3D, are compared with the results of EXCELT module of DRAGON when possible. Chapter 5 discusses computation of tracks in hexagonal assemblies (2D and 3D), by using the above mentioned algorithms. Chapter 6 presents our conclusions.

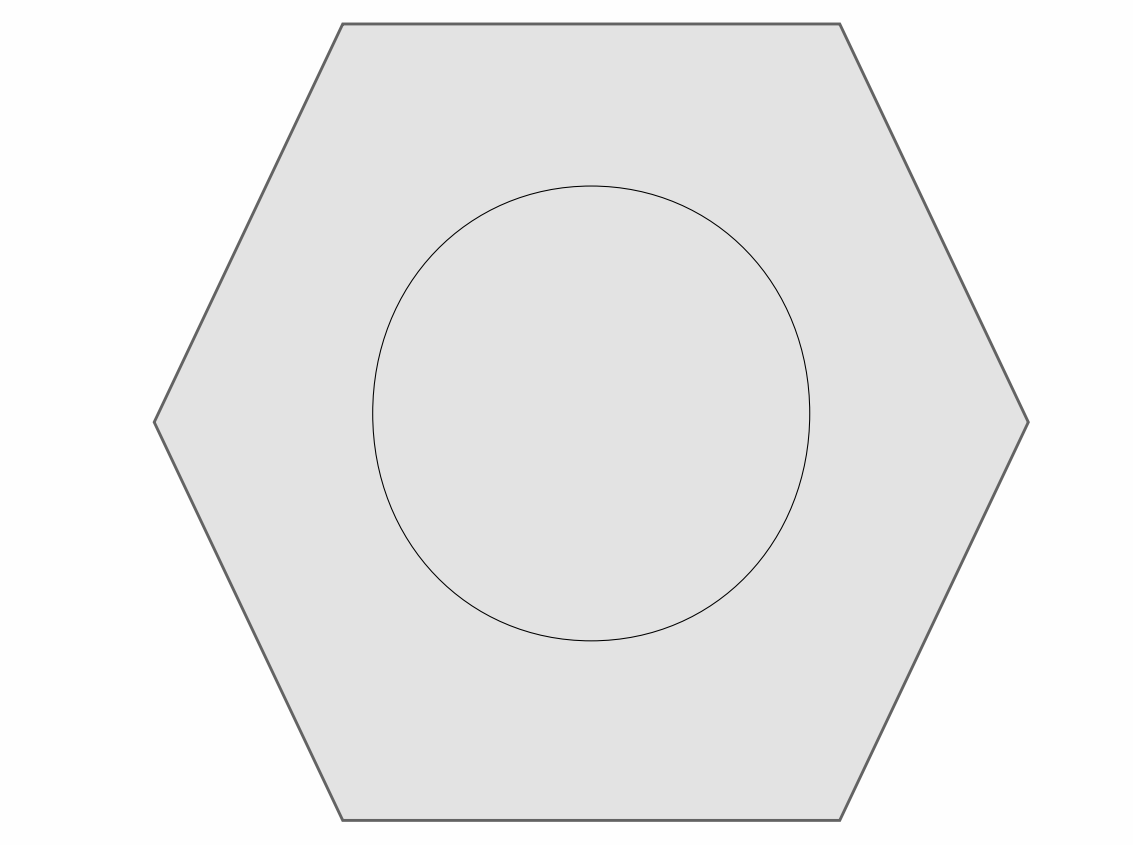


Figure 1.1 Single hexagonal pin cell in 2D

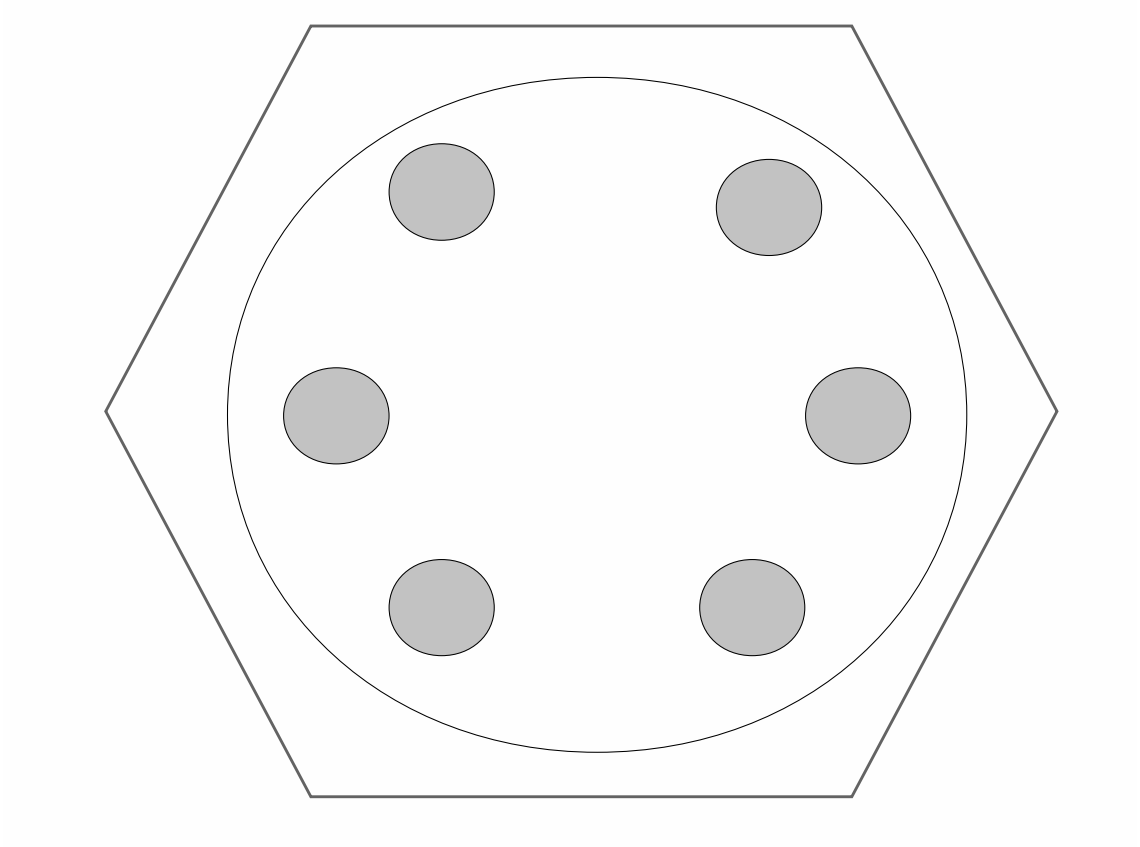


Figure 1.2 Single hexagonal cluster in 2D

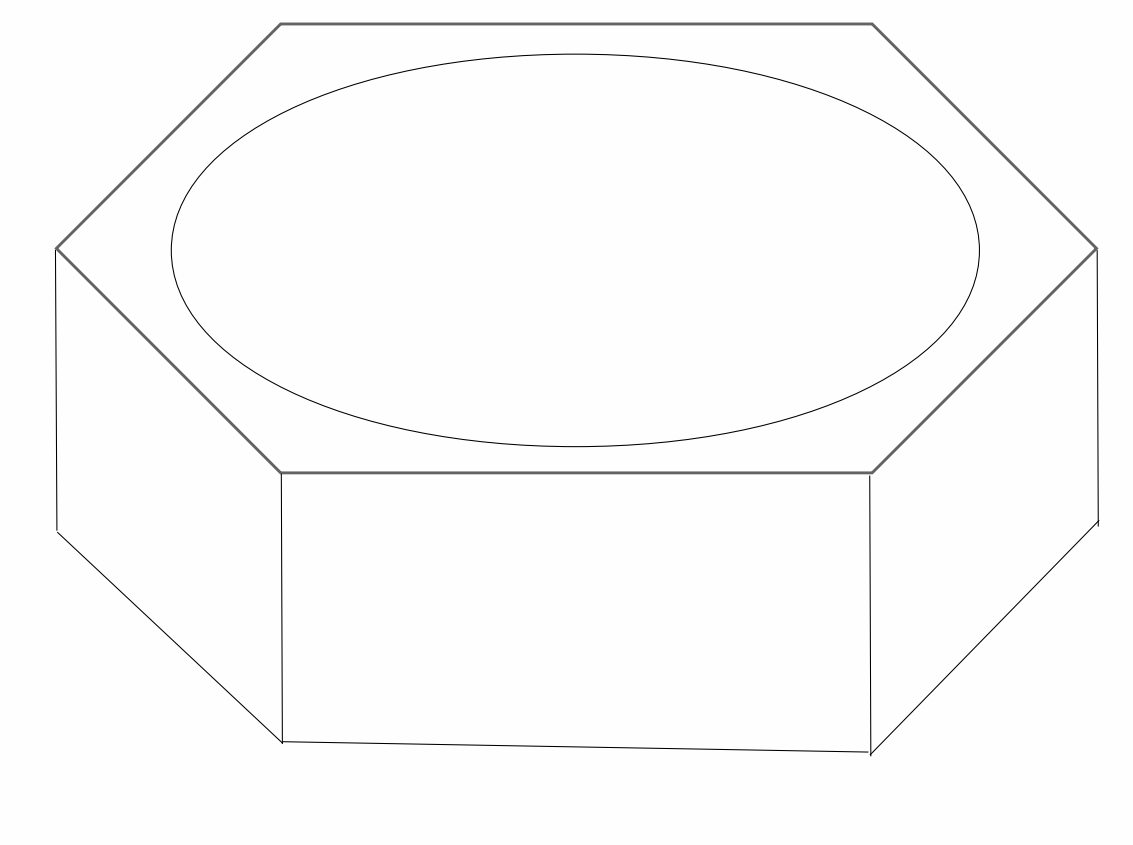


Figure 1.3 Single hexagonal pin cell in 3D

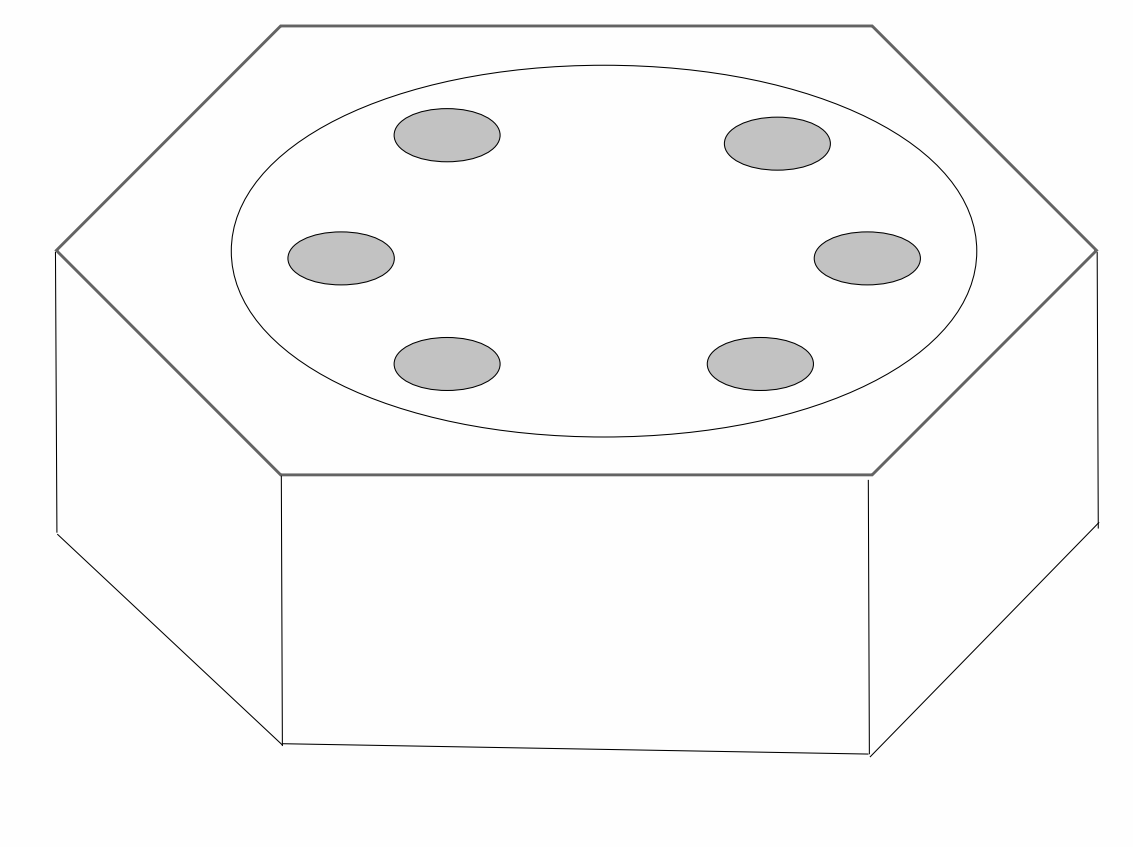


Figure 1.4 Single hexagonal cluster in 3D

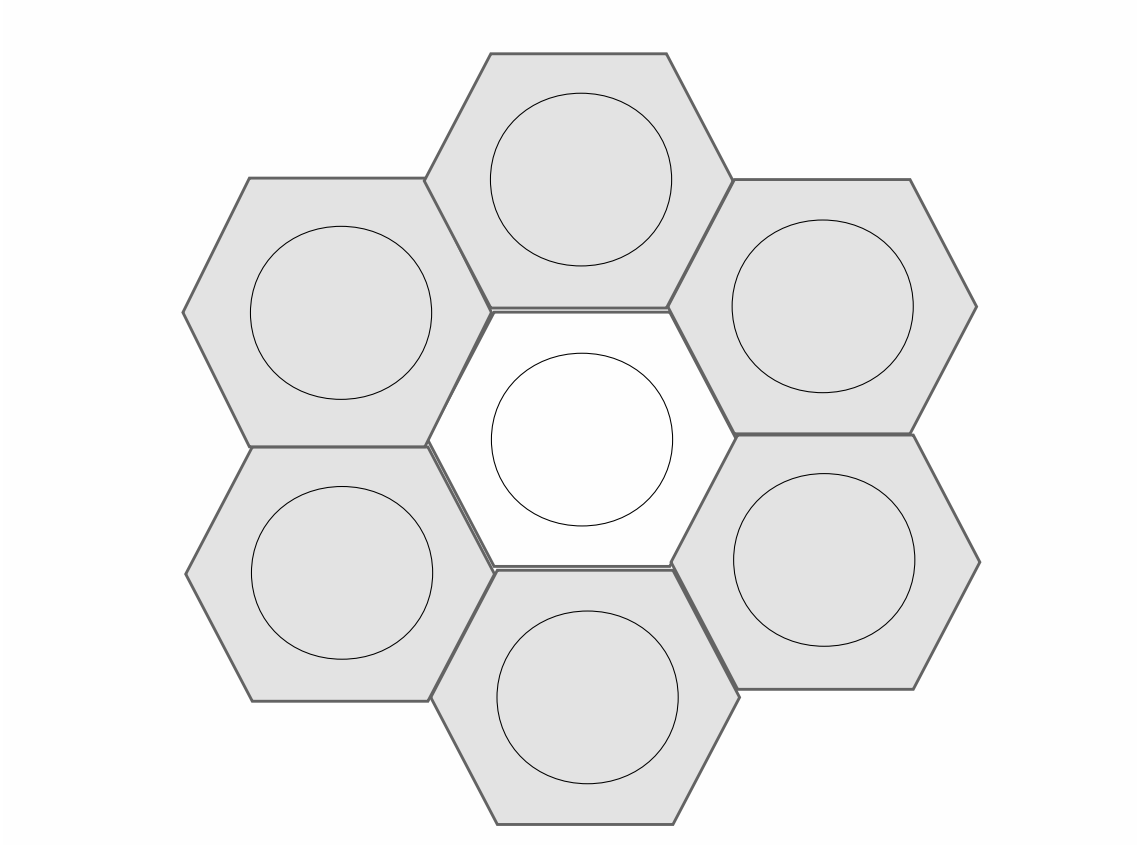


Figure 1.5 Example of a 2D assembly of hexagonal cells

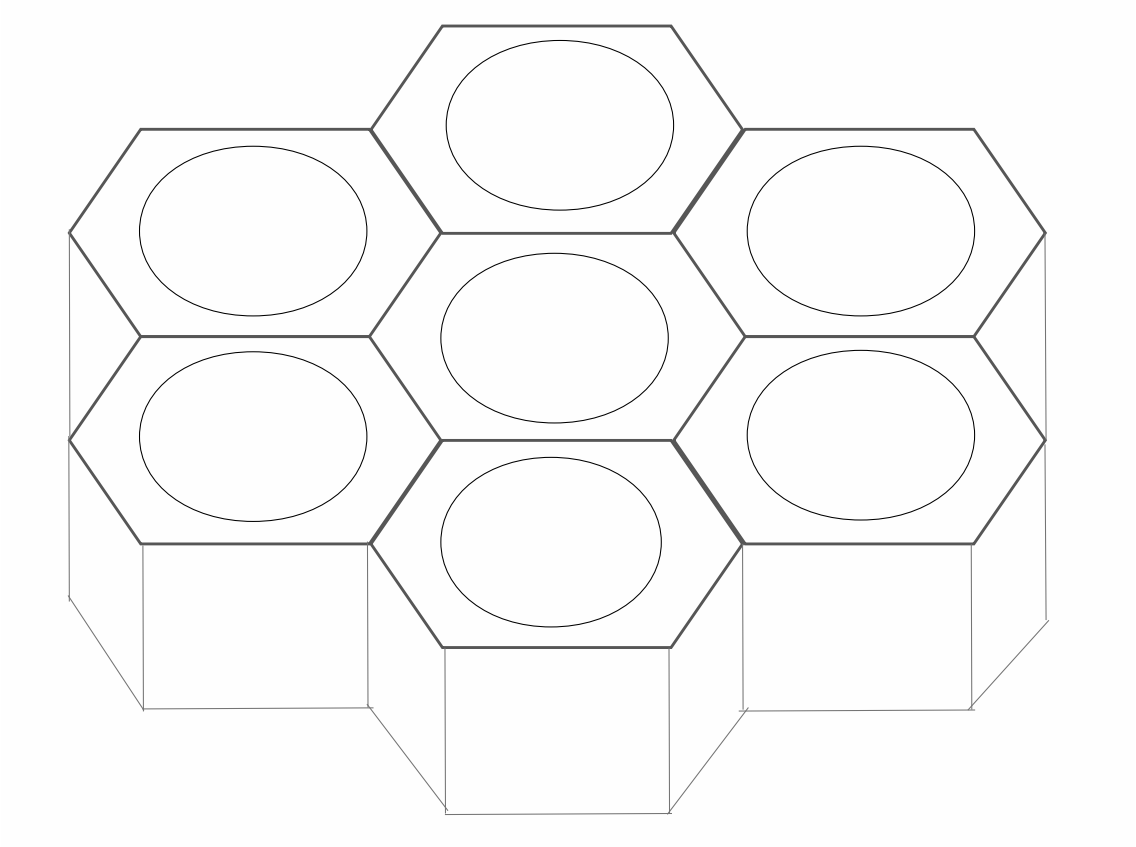


Figure 1.6 Example of a 3D assembly of hexagonal cells

CHAPTER 2

COLLISION PROBABILITY METHOD AND FLUX COMPUTATIONS

2.1 Introduction

To understand the behavior of a nuclear reactor, computation of neutron fluxes in the reactor is required. These fluxes are solutions of neutron transport equation which is derived by considering balance of neutrons in a given reactor volume. It can be written in integro-differential form [13] and integral form. Due to its complexity, this equation can be solved analytically only for some very simple cases. For realistic systems with complex geometry, neutron fluxes can only be obtained using numerical methods.

In reactor physics, it is sufficient to evaluate the mean reaction rates in various regions of a nuclear reactor. It is therefore reasonable to derive the neutron balance equations for the mean values directly, rather than solving the neutron transport equation first and then integrating afterwards. Collision probability method is a fast and reliable tool for the direct evaluation of such mean values.

2.2 Collision Probability Method and Flux computations

There are various collision probability methods, developed over a period of time. We have used Carlvik's method of collision probabilities. This method can consider exact geometrical details of a lattice in two and three dimensions. In this method, one is required to compute collision probabilities from one zone to another. Here the neutron first flight attenuation factor, from one zone to other is integrated over each zone volume while using the ray tracing method. The usual assumption required is flat source assumption over each zone volume. It requires that a zone should be thinner for better accuracy. After obtaining these collision probabilities, neutron fluxes are computed. Here we discuss in general collision probability method, flux computations and computation of collision probabilities.

2.2.1 Collision probability method

The one group integral form of the transport equation [13, 22], with zero incoming current, is

$$\phi(\mathbf{r}) = \int_{4\pi} d^2\Omega \phi(\mathbf{r}, \mathbf{\Omega}) = \int_{4\pi} d^2\Omega \int_0^\infty dR' e^{-\tau(R')} Q(\mathbf{r} - R'\Omega) \tag{2.1}$$

Defining $\mathbf{r}' = \mathbf{r} - R'\Omega$, this equation can be written as

$$\phi(\mathbf{r}) = \int_{4\pi} d^2\Omega \int_0^\infty dR' e^{-\tau(R')} Q(\mathbf{r}') \tag{2.2}$$

where $\tau(R')$ is the optical path length between the points $\mathbf{r}$ and $\mathbf{r}'$ and is given by

$$\tau(R') = \int_0^{R'} ds \Sigma(\mathbf{r} - s\Omega) \tag{2.3}$$

Let us assume that the space is divided into many sub-regions i such that source $Q(\mathbf{r}')$ is assumed to be uniform Q_i in each sub-region i (Fig. 2.1). It is also assumed that cross-sections and fluxes are uniform in the volume i. We get

$$Q(\mathbf{r}') = Q_i = \Sigma_{si}\phi_i + S_i \tag{2.4}$$

Here S_i is the external source, which may include fission source ($\frac{\nu\Sigma_{fi}\phi_i}{K_{eff}}$) one gets

$$\phi(r) = \sum_i \int_{4\pi} d^2\Omega \int_{R_{i-\frac{1}{2}}}^{R_{i+\frac{1}{2}}} dR' e^{-\tau(R')} Q_i \tag{2.5}$$

Integrating over volume j, where ϕ_j is the average flux given by

$$\phi_j = \frac{\int_{Vj} \phi(r) d^3r}{V_j} \tag{2.6}$$

One gets

$$\Sigma_j V_j \phi_j = \sum_i Q_i V_i P_{ij} \tag{2.7}$$

where

$$P_{ij} = \frac{\Sigma_j}{4\pi V_i} \int_{V_i} d^3r \int_{4\pi} d^2\Omega \int_{R_{j-\frac{1}{2}}}^{R_{j+\frac{1}{2}}} dR' e^{-\tau(R')} \tag{2.8}$$

Here P_{ij} is the collision probability for a neutron born in the region i to collide in the j^{th} region. Defining

$$p_{ij} = \frac{P_{ij}}{\Sigma_j} \tag{2.9}$$

Using reciprocity and conservation relations as

$$p_{ij} V_i = p_{ji} V_j \tag{2.10}$$

and

$$\sum_j p_{ij}\Sigma_j = 1 \tag{2.11}$$

One gets

$$\phi_j = \sum_i Q_i p_{ji} \tag{2.12}$$

This expression is used to compute neutron flux distribution in the given geometry. For this purpose one is required to know the fission, scattering, total cross-sections and collision probabilities.

Expressions used for the computation of collision probabilities for 2D and 3D geometries are given below [22].

2.3 Computation of collision probabilities

The expression for the collision probabilities (Eq. 2.8) can be written [22] as

$$P_{ij} = \frac{\Sigma_j}{4\pi V_i}\int_{x\in i} dx \int_{y\in i} dy \int_{R_{i-\frac{1}{2}}}^{R_{i+\frac{1}{2}}} dR \int_{4\pi} d^2\Omega \int_{R_{j-\frac{1}{2}}}^{R_{j+\frac{1}{2}}} dR' e^{-\tau(R')} \tag{2.13}$$

where

$$d^3r = dxdydR \tag{2.14}$$

Let us assume that a line makes an angle α with the x-axis and an angle β with the vertical z-axis. Then integral over all the angles can be expressed as

$$\int d^2\Omega = \int_0^{2\pi} d\alpha \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin\beta d\beta \tag{2.15}$$

Computation of collision probability P_{ij} requires computation of six integrals.

2.3.1 Collision probabilities for 2D geometries

In this case the integrals (Eq. 2.13) over R and R' can be evaluated analytically. By using Bickley functions for β integral, the equations reduce to double integral as will be mentioned below. Bickley functions [2] are defined as

$$Ki_n(x) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^{(n-1)\beta} e^{-x/\sin\beta} d\beta \tag{2.16}$$

For 3D geometries β integration also has to be evaluated numerically, this will be discussed in the next section.

By using Eqs. (2.13) and (2.16) and defining [22]

$$r_{ij} = V_i p_{ij} = V_j p_{ji} \tag{2.17}$$

One gets (when $i \neq j$)

$$r_{ij}^{2D} = \frac{1}{2\pi\Sigma_i\Sigma_j}\int_0^{2\pi} d\alpha \int_{y\in V_i} dy \Big[Ki_3(\tau_{i-\frac{1}{2},j+\frac{1}{2}}) - Ki_3(\tau_{i-\frac{1}{2},j-\frac{1}{2}} - Ki_3(\tau_{i+\frac{1}{2},j+\frac{1}{2}}) + Ki_3(\tau_{i+\frac{1}{2},j-\frac{1}{2}})\Big] \tag{2.18}$$

where $\tau = \tau(\alpha, y) = \Sigma u(\alpha, y)$.

For the region i, the optical path lengths τ_i are distances in 2D (u^i) multiplied by the respective total cross-section Σ_i of the region i. The method of computing tracks u^i, in hexagons, will be discussed in chapters 3 and 5. When tracks in one region are computed, suffix i is dropped.

The first and second boundary, of the track u, of the i^{th} region, are denoted as $(i-\frac{1}{2}, i+\frac{1}{2})$. We define

$$u_1^i = u_{i-\frac{1}{2}} \tag{2.19}$$

and

$$u_2^i = u_{i+\frac{1}{2}} \tag{2.20}$$

Where

$$u^i = (u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}) \tag{2.21}$$

We have

$$\tau_i = \tau_{i-\frac{1}{2},i+\frac{1}{2}} = \Sigma_i u^i \tag{2.22}$$

$$\tau_{ij} = \sum_{m=i+\frac{1}{2}}^{m=j-\frac{1}{2}} \Sigma_m u^m \tag{2.23}$$

or

$$\tau_{i+\frac{1}{2},j-\frac{1}{2}} = \tau_{ij} \tag{2.24}$$

$$\tau_{i-\frac{1}{2},j-\frac{1}{2}} = \tau_{ij} + \tau_i \tag{2.25}$$

$$\tau_{i-\frac{1}{2},j+\frac{1}{2}} = \tau_{ij} + \tau_j + \tau_i \tag{2.26}$$

Here $\tau_i, \tau_j, \tau_{ij}$ are respectively the optical path lengths in i^{th}, j^{th} region and between i and j region as can be seen from Fig. 2.1. The optical path length $\tau(R')$ in Eq. (2.13) can be

written as

$$\tau(R') = (\Sigma_i R + \tau_{ij} + \Sigma_j R')/\sin\beta$$

When the total cross-sections in any of the regions i, j or in both the regions are zero, their respective contributions to the optical path length is zero. The integration of Eq. (2.13), over the region results in the track length of the region. Accordingly, the expressions of the probabilities have either Ki_2 or Ki_1 functions [22].

When $\Sigma_j = 0$, one gets [22]

$$r_{ij}^{2D} = \frac{V_j^{2D}}{2\pi\Sigma_i}\int_0^{2\pi} d\alpha \left[Ki_2(\tau_{i+\frac{1}{2},j-\frac{1}{2}}) - Ki_2(\tau_{i-\frac{1}{2},j-\frac{1}{2}})\right] \tag{2.27}$$

where

$$V_j^{2D} = \int_{R'\in V_j} dR' \int_{y\in V_j} dy \tag{2.28}$$

When $\Sigma_i = \Sigma_j = 0$ we have

$$r_{ij}^{2D} = \frac{1}{2\pi}\int_0^{2\pi} d\alpha \int_{y\in V_i} dy u^i u^j \left[Ki_1(\tau_{i+\frac{1}{2},j-\frac{1}{2}})\right] \tag{2.29}$$

When $i = j$, [22] one gets

$$r_{ii}^{2D} = \frac{1}{\pi\Sigma_i^2}\int_0^{2\pi} d\alpha \int_{y\in V_i} dy \left[Ki_2(0)\tau_{i-\frac{1}{2},i+\frac{1}{2}} + Ki_3(\tau_{i-\frac{1}{2},i+\frac{1}{2}}) - Ki_3(0)\right] \tag{2.30}$$

when $\Sigma_i = 0$, one gets

$$r_{ii}^{2D} = \frac{1}{4}\int_0^{2\pi} d\alpha \int_{y\in V_i} dy (u^i)^2 \tag{2.31}$$

$$= \frac{Ki(0)}{2\pi}\int_0^{2\pi} d\alpha \int_{y\in V_i} dy (u^i)^2 \tag{2.32}$$

2.3.2 Collision probabilities for 3D geometries

In the Eq. (2.13), the integral over R' is in the direction of the track [22]. The first and second boundary of the i^{th} region are denoted as $(i-\frac{1}{2}, i+\frac{1}{2})$. Here we use the notations

$$d_{31}^i = R'_{i-\frac{1}{2}} \tag{2.33}$$

when $n \neq j$

$$d_{32}^i = R'_{i+\frac{1}{2}} \tag{2.34}$$

$$d_3^i = R'_{i+\frac{1}{2}} - R'_{i-\frac{1}{2}} \tag{2.35}$$

Optical path lengths τ' are defined as distances (R' or d_3^i) in 3D multiplied by respective total cross-sections (Fig. 2.2). One defines

$$\tau'_i = \Sigma_i d_3^i \tag{2.36}$$

$$\tau'_{ij} = \sum_{m=i+\frac{1}{2}}^{m=j-\frac{1}{2}} \Sigma_m d_3^m \tag{2.37}$$

The track distances d_3^i will be computed in chapter 4 (for one hexagon) and in chapter 5 (for seven hexagons).

When $i \neq j$, one gets [22] by simplifying (Eq. 2.13)

$$r_{ij}^{3D} = \frac{1}{4\pi\Sigma_i\Sigma_j} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_j} dy$$
$$\left[1 - \exp(-\tau'_{i-\frac{1}{2},i+\frac{1}{2}})\right] \exp(-\tau'_{i+\frac{1}{2},j-\frac{1}{2}}) [1 - \exp(\tau'_{j-\frac{1}{2},j+\frac{1}{2}})\Big] \tag{2.38}$$

when $\Sigma_i = 0$, one gets

$$r_{ij}^{3D} = \frac{1}{4\pi\Sigma_j} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_j} dy \left[\exp(-\tau'_{i+\frac{1}{2},j-\frac{1}{2}})][1 - \exp(\tau'_{j-\frac{1}{2},j+\frac{1}{2}})\right] d_3^i \tag{2.39}$$

when $\Sigma_j = 0$, one gets

$$r_{ij}^{3D} = \frac{1}{4\pi\Sigma_i} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_j} dy \left[\exp(-\tau'_{i+\frac{1}{2},j-\frac{1}{2}})][1 - \exp(\tau'_{i-\frac{1}{2},i+\frac{1}{2}})\right] d_3^j \tag{2.40}$$

when both $\Sigma_i = 0$ and $\Sigma_j = 0$ [22]

$$r_{ij}^{3D} = \frac{1}{4\pi} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_j} dy [\exp(-\tau'_{i+\frac{1}{2},j-\frac{1}{2}})] d_3^i d_3^j \tag{2.41}$$

When $i = j$, one gets

$$r_{ii}^{3D} = \frac{1}{2\pi\Sigma_i^2} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_i} dy \left[\tau'_{i-\frac{1}{2},i+\frac{1}{2}} - (1 - \exp(-\tau'_{i-\frac{1}{2},i+\frac{1}{2}})\right] \tag{2.42}$$

when $\Sigma_i = 0$, one gets [22]

$$r_{ii}^{3D} = \frac{1}{4\pi} \int_0^{4\pi} d^2\Omega \int_{x\in V_i} dx \int_{y\in V_i} dy (d_3^i)^2 \tag{2.43}$$

Here $d_3^i(\Omega, x, y)$ is the 3D track length in the i^{th} region. These expressions require computation of integrals and computation of track lengths u^i in 2D and d_3^i in 3D . First we will discuss computation of integrals and then computation of track lengths.

2.4 Computation of Integrals

All the above mentioned expressions for collision probabilities (Eqs. 2.18 to 2.43) have integrals. These integrals are approximated by various numerical methods. We have used equal weights quadrature methods for computing all the integrals. Here each of the integrals can be expressed as

$$\int_a^b P(x)dx = w \sum_n P(v_n) \tag{2.44}$$

where v_n are the integration points and w is the integration weight. The limits of integration are a and b.

2.4.1 2D geometries

The 2D collision probability expressions Eqs. (2.18 to 2.32) have double integrals. One integral is over angle α and the other integral is over space y.

When these integrals are approximated numerically, the respective weights w'_A, w_S for angle and space integration and the integration points v are selected in the folowing way. The angle integration of these equations, over α, varies from $a = 0$ to $b = \pi$, since the contribution from π to 2π will be associated with the probability P_{ji} which are symmetric to P_{ij}. Here $2N$ number of points are selected from $a = 0$ to $b = \pi$. We get

$$w'_A = \frac{(b-a)}{2N} = \frac{\pi}{2N} \tag{2.45}$$

and

$$v'_n = (n' - \frac{1}{2})w'_A = (n' - \frac{1}{2})\frac{\pi}{2N} \tag{2.46}$$

For the y integral, a trapezoidal quadrature set is selected [22]. Hexagons are supposed to be surrounded by a circle of radius R. For one hexagon $R = d$, where d is side of the hexagon. The limits of space or y integral are $a = -R$ and $b = R$. Number of equal subdivisions selected are L. Spacing δ between the two points or weight w_y is given by

$$\delta = w_y = \frac{(b-a)}{L} = \frac{2R}{L} \tag{2.47}$$

Tracking density selected is T. The number of selected points L are related to T by

$$L = 2RT + 1 \tag{2.48}$$

and

$$v_l = (l - \frac{1}{2})\delta \tag{2.49}$$

or

$$v_l = (l - \frac{1}{2})\frac{2R}{L} \tag{2.50}$$

2.4.2 3D geometries

In the case of 3D hexagon cells, there is a difference between the NXT and EXCELT modules, in the numerical quadratures used for the computation of integrals. We will describe these differences in short [22].

For the module NXT, the integral over the solid angle is discretized using equal weight EQ_N quadrature technique [6]. Each direction Ω_q can be written in terms of its direction cosines:

$$\Omega_q = (v_{xq}, v_{yq}, v_{zq}) \tag{2.51}$$

such that

$$v_{xq}^2 + v_{yq}^2 + v_{zq}^2 = 1 \tag{2.52}$$

The $d^2\Omega$ integration can be written as a double angular integration over α and β as given in equation (2.15). Here

$$0 \leq \alpha \leq 2\pi \tag{2.53}$$

$$0 \leq \beta \leq \pi/2 \tag{2.54}$$

For a N^Ω solid angle quadrature, $N^\Omega(N^\Omega + 2)/2$ weights and points are selected in the upper sphere. The integrals over dx and dy are discretized over a plane normal to the direction Ω. The limits of integration over these angles are specified in such a way that they are independent of the specific direction. A sphere of radius R is supposed to be surrounding the complete geometry. The integration limits over x and y are given by

$$-R \leq x \leq R \tag{2.55}$$

$$-R \leq y \leq R \tag{2.56}$$

Tracking density of T is equal to selecting N^2 spatial points such that

$$N = (2\sqrt{T}R) + 1 \tag{2.57}$$

The spacing between the tracks is given by

$$\delta = \frac{2R}{N} \tag{2.58}$$

The quadrature weights and points are given by

$$w_x w_y = \delta^2 \tag{2.59}$$

$$v_l^x = (l - \frac{1}{2})\delta \tag{2.60}$$

$$v_n^y = (n - \frac{1}{2})\delta \tag{2.61}$$

The locations of (x, y) points are defined arbitrarily w.r.t. the direction Ω, which represents the z axis. In the DRAGON code, the surface integral over a hexagon cell is invariant under a rotation of integral plane. For each direction Ω three integration planes are selected and tracked successively. The weight associated with each direction is reduced by a factor of 3. Details are given in [22].

In the EXCELT module computations for 3D hexagons is performed in a slightly different way. Here EQ_N integration over solid angle $d^2\Omega$ is replaced by double integral over $d\alpha d\beta$ ($0 \leq \alpha \leq 2\pi$ and $0 \leq \beta \leq \pi/2$). The surface integral over $dxdy$ is replaced by $dudz$ where du integral is located in the $2D$ $(x - y)$ plane and dz integral defines between plane and the z-axis, details are given in the references ([22, 30]).

2.4.3 Numerical expression of collision probabilities

The integrals for 2D and 3D collision probabilities are given in Eqs.(2.18) and (2.38). Numerically they can be expressed as

$$r_{ij}^{2D} = w'_A w_y \frac{1}{2\pi\Sigma_i\Sigma_j} \sum_{l'} \sum_{n}$$
$$\Big[Ki_3({}^{l',n}\tau_{i-\frac{1}{2},j+\frac{1}{2}}) - Ki_3({}^{l',n}\tau_{i-\frac{1}{2},j-\frac{1}{2}}) - Ki_3({}^{l',n}\tau_{i+\frac{1}{2},j+\frac{1}{2}}) + Ki_3({}^{l',n}\tau_{i+\frac{1}{2},j-\frac{1}{2}})\Big] \tag{2.62}$$

$$r_{ij}^{3D} = w'_A w'_B w_x w_y \frac{1}{4\pi\Sigma_i\Sigma_j} \sum_{l'} \sum_{n'} \sum_{l} \sum_{n}$$
$$\left[1 - \exp(-^{l',n',l,n}\tau'_{i-\frac{1}{2},j+\frac{1}{2}})\right] \exp(-^{l',n',l,n}\tau'_{i+\frac{1}{2},j-\frac{1}{2}})[1 - \exp(^{l',n',l,n}\tau'_{i-\frac{1}{2},j+\frac{1}{2}})\Big] \tag{2.63}$$

All of these integrals will be computed by the equal weight quadrature method, mentioned above.

2.5 Expressions for volume and surfaces

2.5.1 Numerical computation of volume and surfaces

We can write volume and surface integrals as

$$V_j^{2D} = \int d\alpha \int_{y \in V_j} dy u^j(\alpha, y) \tag{2.64}$$

or

$$V_j^{2D} = w'_A w_y \sum_{l',n} u^j_{l',n} \tag{2.65}$$

and

$$V_j^{3D} = \int d^2\Omega \int dx \int dy d_3^j(\Omega, x, y) \tag{2.66}$$

or

$$V_j^{3D} = w'_A w'_B w_x w_y \sum_{l',n',l,n} d^j_{3,l',n',l,n} \tag{2.67}$$

Similarly surface integrals can be expressed as

$$S_j^{2D} = \int d\alpha \int_{y \in V_j} dy \tag{2.68}$$

or

$$S_j^{2D} = w'_A w_y \sum_{n} \delta_{n,j} \tag{2.69}$$

When $n \neq j$

$$\delta_{n,j} = 0 \tag{2.70}$$

and when $n = j$

$$\delta_{n,j} = 1 \tag{2.71}$$

$$S_{3D}^{j} = \int d^2\Omega \int dx \int dy \tag{2.72}$$

or

$$S_{3D}^{j} = w'_A w'_B w_x w_y \sum_{l,n} \delta_{l,n,j} \tag{2.73}$$

When $l, n \in\neq V_j$

$$\delta_{l,n,j} = 0 \tag{2.74}$$

When $l, n \in V_j$

$$\delta_{l,n,j} = 1 \tag{2.75}$$

2.5.2 Analytical computations of volume and surfaces

In the NXT module, we are required to compute the volume of all the regions and surface areas of all the surfaces. Since all the seven hexagons are identical, we will compute volume and surface areas for one hexagon only. In the case of one hexagon, two cases are considered: a) one hexagon with one pin and b) one hexagon with a cluster of six pins. In all the cases, side of one hexagon is denoted by d. In the case of 3D, height of the hexagons is denoted by h. When it is a 2D case $h = 1$ is considered. Radius of a single pin is denoted as r. All the pins have same radius.

Surface areas S_h of bottom and top surfaces of a hexagon are given by

$$S_h = 3\sqrt(3)d^2/2 \tag{2.76}$$

a) Hexagon cell with one pin:

In this case (Figs. 2.3) there are 2 regions, region 1 of pin and region 2 of the moderator region.

Area of one pin is given by

$$S_{p_1} = \pi r^2 \tag{2.77}$$

Top and bottom surfaces area of the moderator are given by

$$S_{m_2} = S_h = 3\sqrt(3)d^2/2 - \pi r^2 \tag{2.78}$$

b) Hexagon cell with a cluster of six pins:

When there is a cluster of six pins (each having radius r) (Figs. 2.3) these pins are supposed to be surrounded by moderator. In this case, the moderator is divided into

two regions, one region is with radius R_1 and the outer region is hexagon region. Total number of regions in this case are 3. Top and bottom surface areas of the moderator region 2, containing cluster of six pins are given by

$$S_{m_2} = \pi R_1^2 - 6\pi r^2 \tag{2.79}$$

Here R_1 is selected such that it lies in the hexagon ($R_1 \leq H$), where $H = d\sqrt{3}/2$. (EXCELT modules requires that all the pins be surrounded by R_1). Top and bottom surface areas of the outer moderator region 3 are given by

$$S_{m_3} = 3\sqrt{(}3)d^2/2 - \pi R_1^2 \tag{2.80}$$

Volumes V are given by the respective surfaces S multiplied by the height (in 2D case $h = 1$ is considered) and they are given by

$$V = Sh \tag{2.81}$$

Side surface areas S_s are the same for all the sides and they are given as

$$S_s = dh \tag{2.82}$$

To validate the volumes V and surface areas S, these are compared in the NXT module, for all the volumes and for all the surfaces, with the numerically computed values. This will be mentioned in chapters 3 to 5. Such computations are done along with the computation of track lengths.

2.6 Computation of the required track lengths

In the method of collision probabilities, the given geometry is considered to be covered by various parallel lines having tracking density of T/cm in 2D case and T/cm^2 in 3D case. For each direction of tracking lines, track lengths in all the regions are computed. Number of such directions selected are N. In the EXCELT module, $d^2\Omega$ integral is divided into two integrals, so the number of tracking directions and tracking densities are denoted as $N1$, $N2$, $T1$ and $T2$ respectively. Results will be given for various values of N and T in chapters 3 to 5.

The respective $2D$ and $3D$ track lengths $u^j(\alpha, y)$ and $d_3^j(\Omega, x, y)$ depend on the direction Ω and the values of the spatial co-ordinates (x, y). If we divide the geometry in various regions then the length of a track depends on the region and surface numbers from which the

track starts. In the next section we discuss the method of region and surface numbering.

Initially tracks are computed in hexagons and then NXT module computes tracks in the pins in the hexagon cells.

2.6.1 Numbering of hexagons and pins

In the NXT module region numbers are denoted as positive numbers. For hexagons the numbering start from the central cell. We name this variable as N. Here N = 1 represents the central cell. Outer hexagonal cells are numbered in the anti clockwise direction with increasing angle with respect to the x-axis. X-axis is considered parallel to the base of the central hexagonal cell. The (x, y) co-ordinates of the center of hexagons are with respect to the center of the central hexagonal cell. For seven hexagons the region numbers are from 1 to 7. This is shown in Figs. 2.3 and 2.8.

In a hexagon cell with pins, numbering starts from numbering of pins first and then it increases for the moderator region. This numbering is the same for 2D and 3D cases. If there is a cluster of pins and the pins are placed at equal distance from the center then the numbers of all the pins remain same. When there is one pin the region numbers vary from 1 to 2, for each hexagon cell. In the case of clusters the region numbers vary from 1 to 3 for the hexagon cell. In the case of seven hexagon cells with each cell having one pin, the region numbers vary from 1 to 14. This is shown in Figs. 2.3 to 2.8.

2.6.2 Numbering of surfaces

In the NXT module, numbering of surfaces is denoted as negative numbers. Initially side surfaces are numbered, then numbering of bottom and top surfaces is done.

2.6.3 Numbering of side surfaces

These numbers will be common for 2D and 3D cases, this can be seen from these figures Figs. 2.5 and 2.10. For one hexagon the side surface numbers vary from -1 to -6 Fig. 2.5. In the case of seven hexagons, the side surface numbers are -1 to -18 Fig. 2.10.

2.6.4 Numbering of bottom and top 3D hexagon surfaces

Numbers of bottom and top surfaces (3D) depend upon the number of pins and number of hexagons. In the case of one hexagon with one pin, there are 2 regions so there are 2 bottom and 2 top surfaces. The total number of surfaces vary from -1 to -10. In the case of cluster of pins in one 3D hexagon, there are 3 regions, the surface number vary from -1 to -12. In the case of seven hexagons (with one pin in each of the hexagons), there are 2 regions in

each of the hexagons. The number of bottom and top surfaces are 2x14. In this case, surface numbers vary from -1 to -46. This is shown in Figs. 2.5 to 2.10.

2.6.5 Tracks inside cylindrical pins

The NXT module computes tracks inside cylindrical pins, then these tracks are subtracted from the tracks computed in a hexagon, to get the tracks in the moderator region surrounding the cylindrical pins. In this way, all the tracks needed for the 2D and 3D hexagon cell computations are known. These are used to compute collision probabilities.

Once collision probabilities are obtained, these are used to compute flux distributions inside lattice cells by using Eq. (2.12).

2.7 Lattices analyzed

For testing the results so obtained, various lattices have been analyzed and average neutron fluxes are compared, when possible, with the EXCELT module of the code DRAGON. All the results have also been verified by plotting them (NXT module does this plotting by using Matlab). Results are computed and compared in one group, for the following lattice cells

a) Single hexagonal pin cells in 2D and 3D.

b) Assembly of seven hexagonal single pin cells in 2D and 3D.

c) Single hexagonal cell with a cluster of pins in 2D and 3D. EXCELT module of the DRAGON code can compute fluxes in a single hexagonal cell with a cluster of pin cells in 2D and not in 3D.

In the next chapter we will present our tracking algorithm for one 2D hexagonal cell.

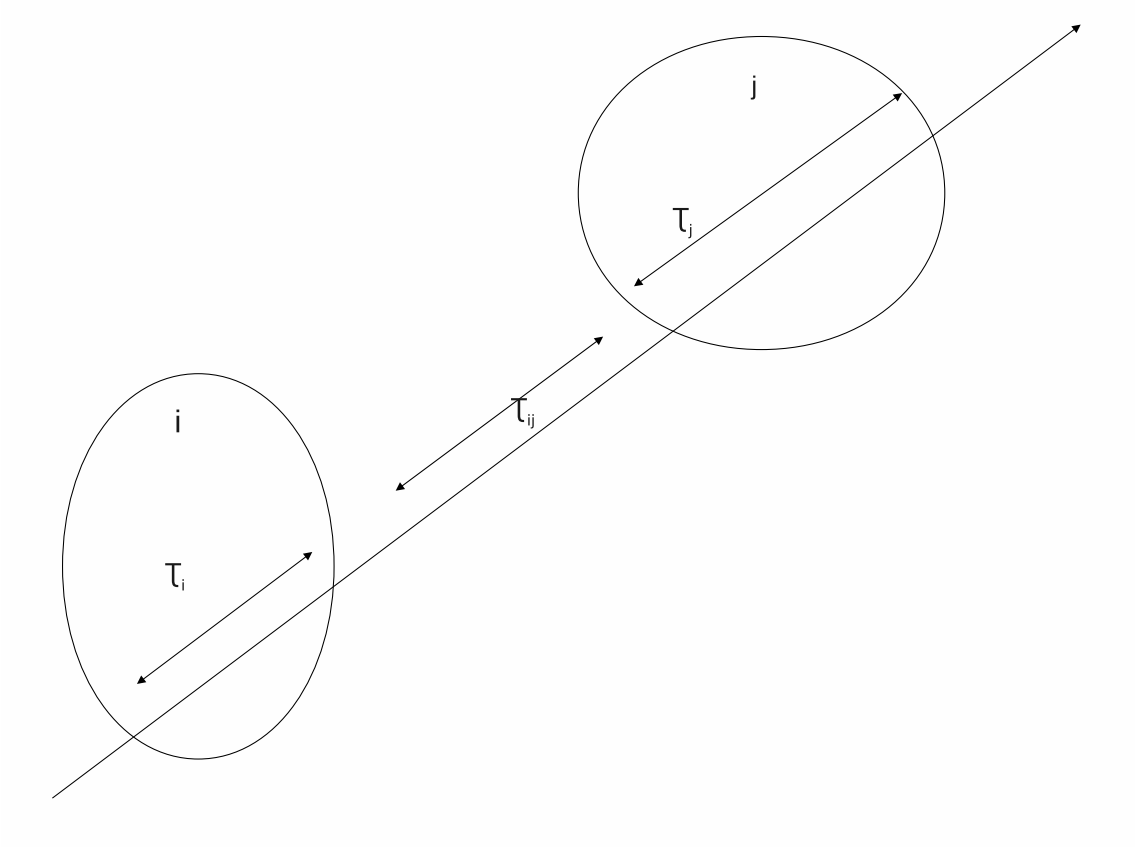


Figure 2.1 Tracks in 2D geometries

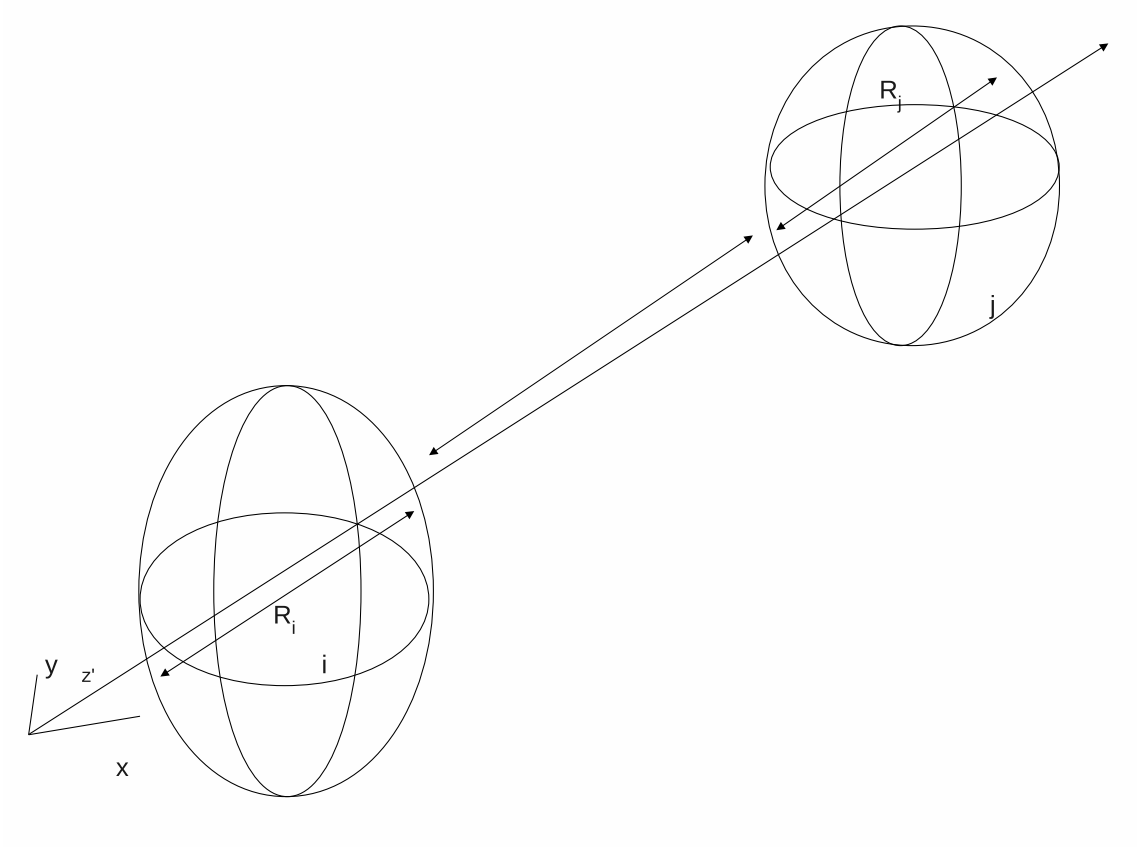


Figure 2.2 Tracks in 3D geometries

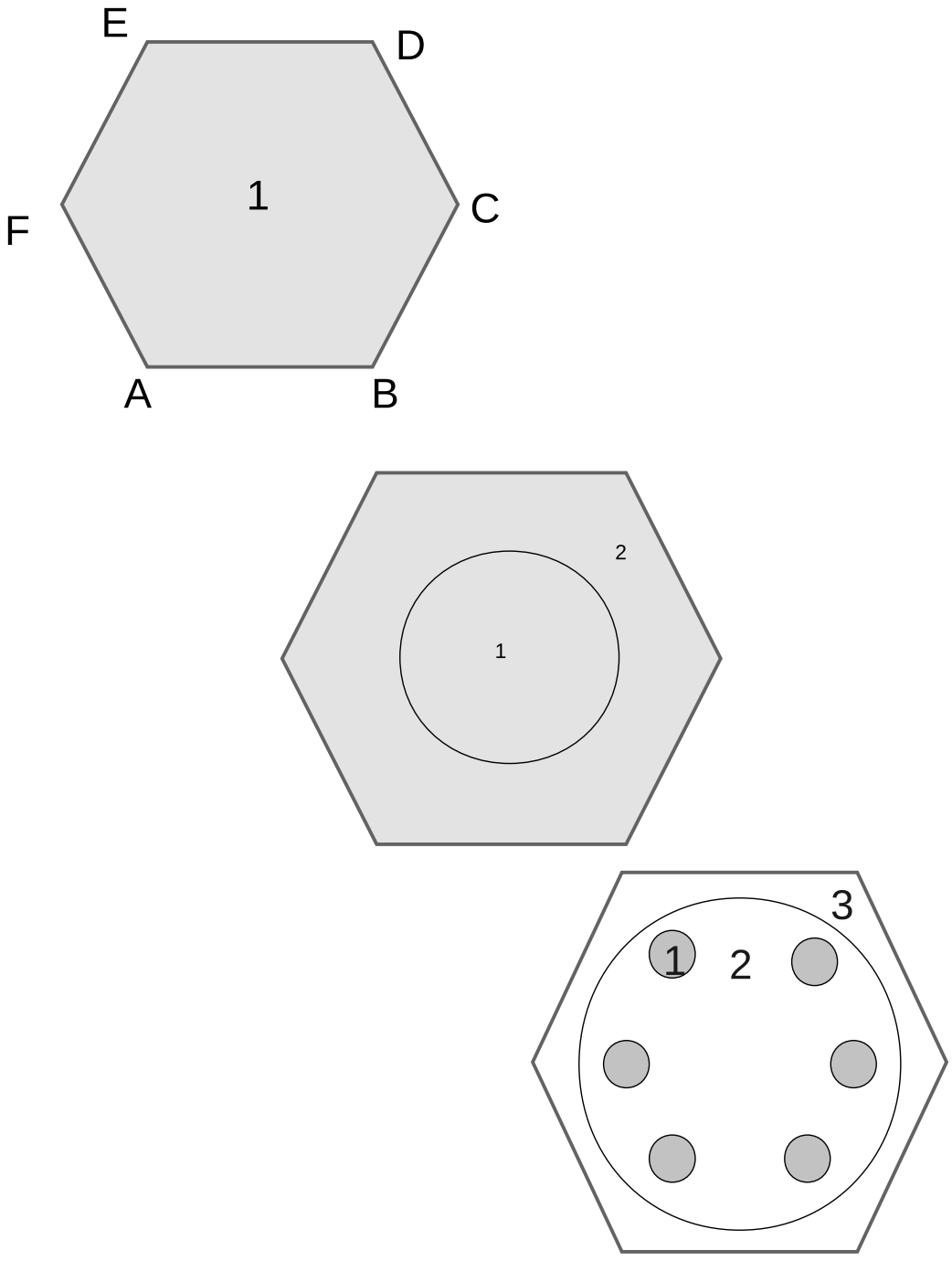


Figure 2.3 Numbering of regions in 2D hexagon cells

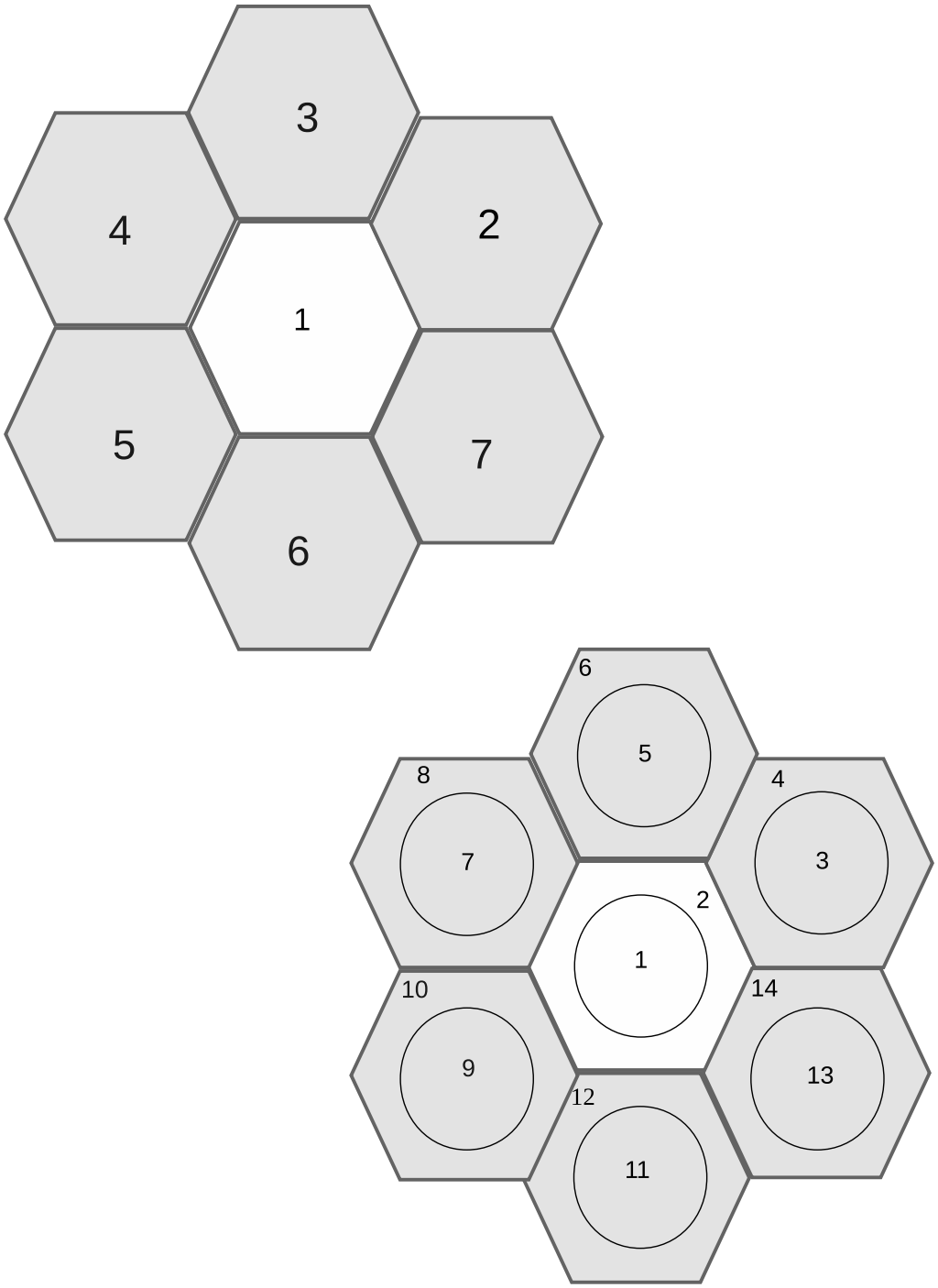


Figure 2.4 Numbering of regions in 2D seven hexagon cells

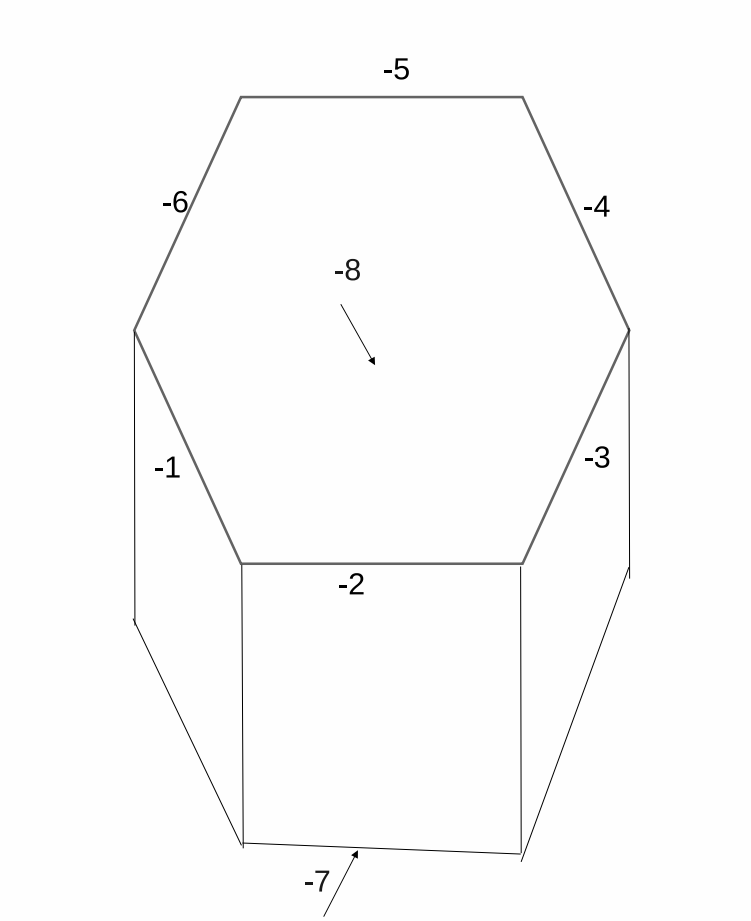


Figure 2.5 Numbering surfaces of a 3D hexagon

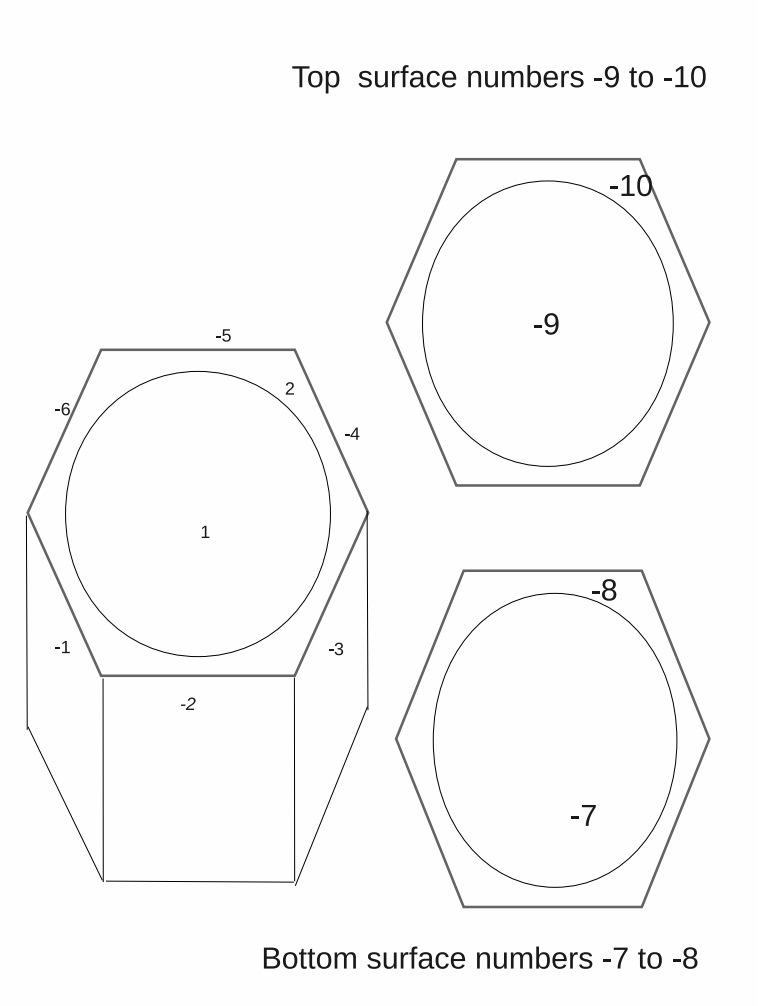


Figure 2.6 Numbering of regions and surfaces of a 3D hexagon cell with one pin

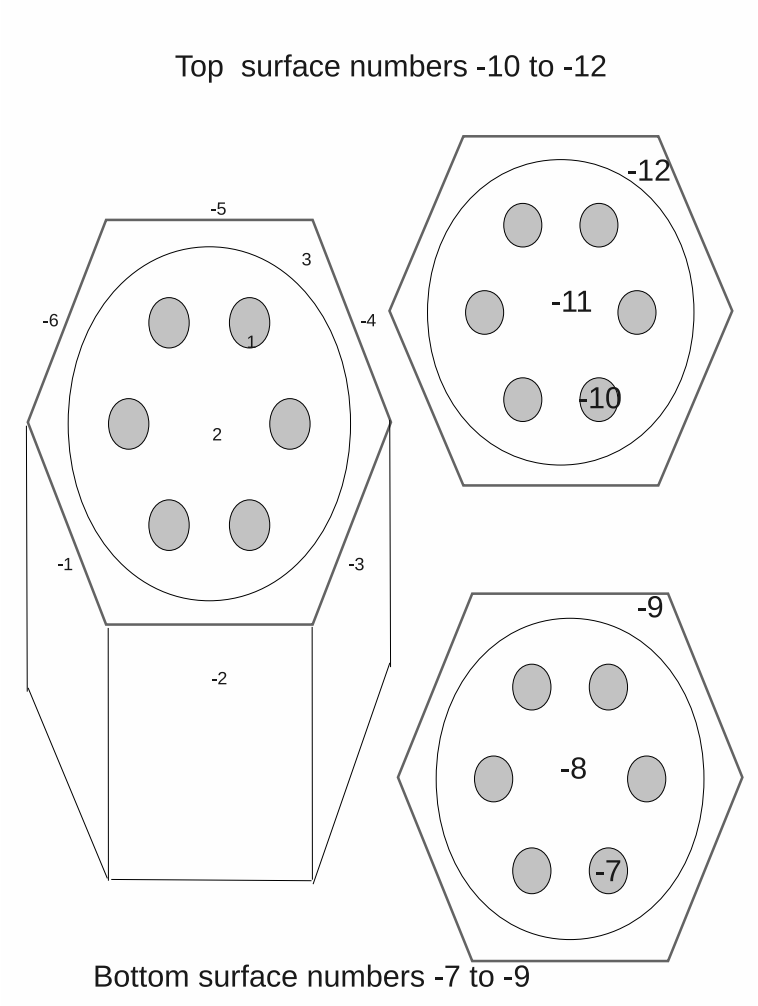


Figure 2.7 Numbering of regions and surfaces of a 3D hexagon cell with a cluster of six pins

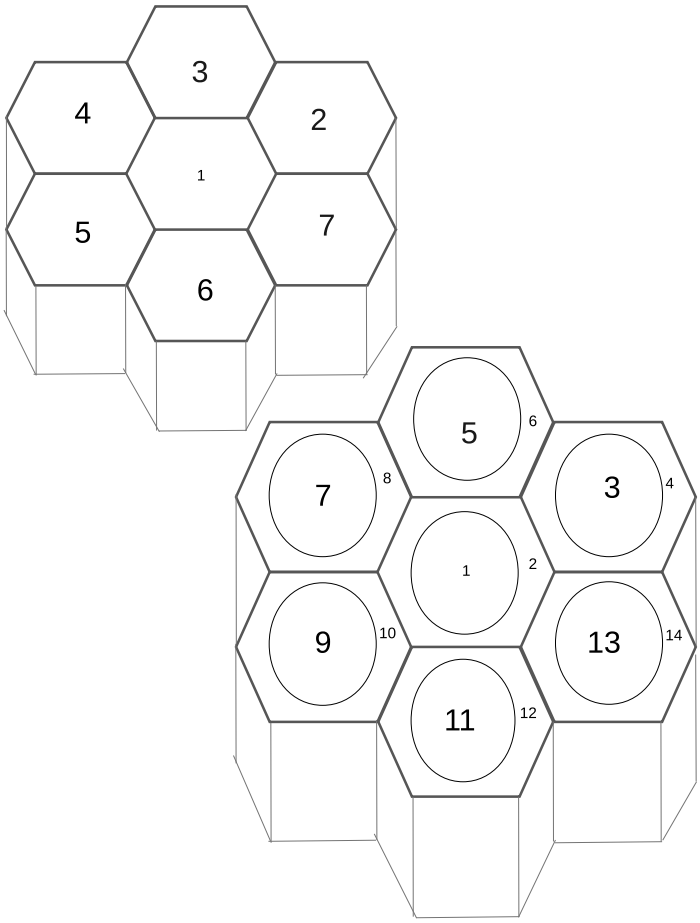


Figure 2.8 Numbering of regions of seven 3D hexagon cells

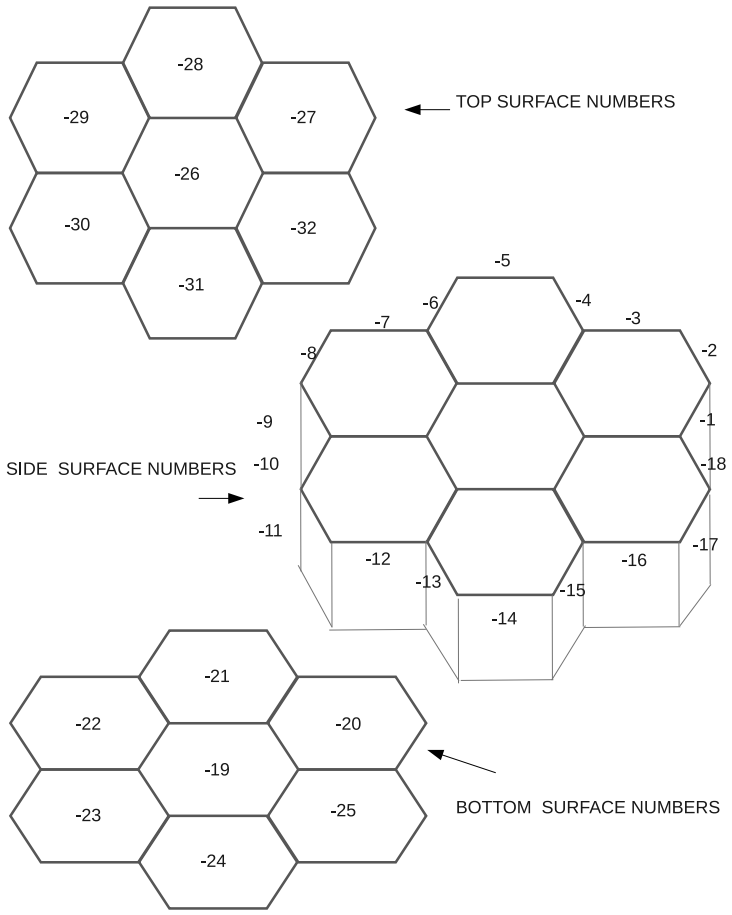


Figure 2.9 Numbering of surfaces for seven 3D hexagons

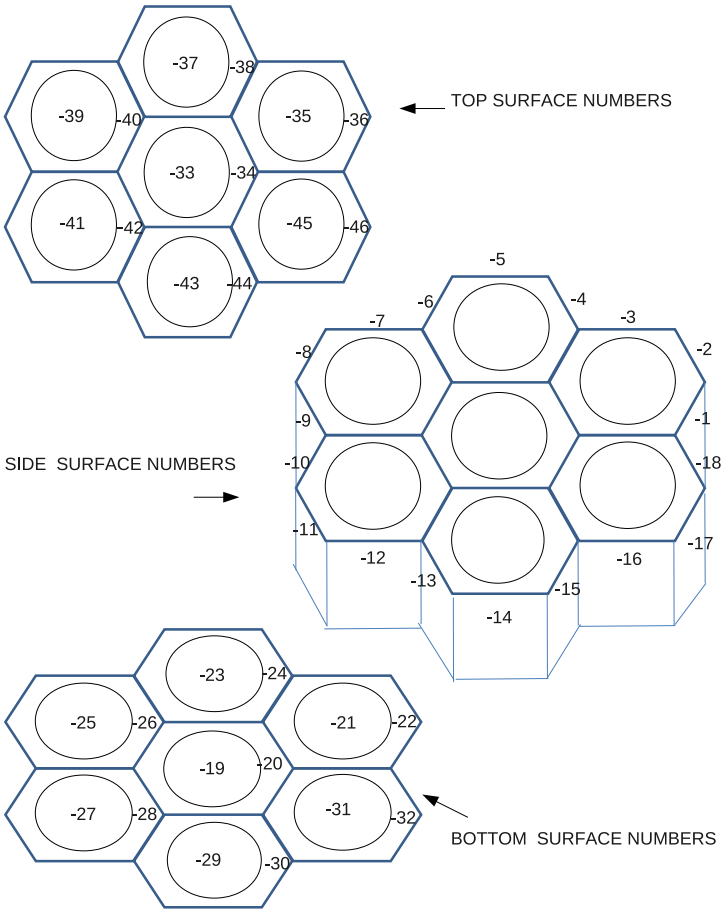


Figure 2.10 Numbering of surfaces of seven 3D hexagon cells (each with one pin)

CHAPTER 3

TRACKING ALGORITHM AND COMPUTATION OF FLUXES IN A 2D HEXAGON CELL

3.1 Introduction

In this chapter we will describe the method of computing tracks in a two dimensional (2D) hexagon. The NXT module [23] of the code DRAGON (Version 3.06) computes tracks in a cylindrical pin or a cluster of pins (depending on the case) in the hexagon cell. These tracks are then used to compute collision probabilities, where these probabilities are used to compute fluxes in the given geometry. We will first describe the tracking algorithm and then we will present the results obtained by using this approach.

3.2 Tracking algorithm in two dimensions (2D)

In this section we will describe how the tracks are computed for one 2D hexagon. These tracks will be used to compute 3D tracks, this will be mentioned in chapter 4. These tracks will also be used to compute tracks in the case of multi hexagonal assemblies (in 2D and 3D). This will be mentioned in chapters 5.

3.2.1 Checking before computations

For computing tracks in one hexagon, we are required to compute intersection points of one line with the sides of the hexagon as can be seen from Fig. 3.1. Before computing any track, first we will check if this line intersects a circle, which surrounds the hexagon as shown in Fig. 3.2. Equations for checking of intersection points of a line with a circle are given in Appendix A. Only when this line intersects this circle that further computations will be done.

3.2.2 Equation for intersection points of two lines

One hexagon, in 2D, can be represented by six lines, that pass through the vertices of the hexagon this can be seen from Fig. 3.3. Let us denote the vertices of the hexagon by (A, B, C, D, E, F) as can be seen in this figure. If the hexagon is crossed by this line, then the line will intersect only two lines out of six lines defining the hexagon. For computing tracks in the hexagon, we have to compute intersection points of the given line with all these

six lines passing through these vertices. While doing the computations, we will consider one side at a time. Let us assume that the given line intersects the side CD. Let the co-ordinates of the points C and D be $C(x_c, y_c)$ and $D(x_d, y_d)$. Let us assume that the line passing through CD, makes an angle λ with the x -axis. A line which makes an angle λ and passes through the point C will pass through D also.

Consider that the line for which tracks lengths are required to be computed makes an angle α with the x-axis and intercepts y-axis at a distance of y_0. Computation of intersection points of two lines, is done in the following way.

Equation of the given tracking line (Fig. 3.3) is

$$y = Mx + y_0 \tag{3.1}$$

where $M = \tan\alpha$

The given line passes through the point (x_a, y_a), we compute y_0 as

$$y_0 = y_a - Mx_a \tag{3.2}$$

Equation of another line which makes an angle λ with the x-axis and passes through the vertex $C(x_c, y_c)$ of the hexagon is

$$y - y_c = m(x - x_c) \tag{3.3}$$

where $m = \tan\lambda$

Points of intersections (x, y) are given by

$$x = \frac{(y_c - mx_c - y_0)}{(M - m)} \tag{3.4}$$

$$y = Mx + y_0 \tag{3.5}$$

Now we want to know, if these intersection points (x, y) are inside the hexagon. For this purpose we will compare x with the x co-ordinates of the points C and D, which are respectively x_c and x_d. If x lies between the points x_c and x_d, it means that it intersects the hexagon side CD and this point is considered.

This process is repeated for all the lines passing through the vertices of the hexagon, which represent the sides of the hexagon. It can be seen from Fig. 3.3 that a line will cut only two sides out of six sides of the hexagon. Let us denote these two side surfaces as 1 and 2.

We denote the two intersection points with the hexagonal surfaces as (x_1, y_1) and (x_2, y_2).

Both the points of intersections $((x_i, y_i), i = 1, 2)$ are stored. Now we want to know which side of the hexagon is intersected first. For this purpose, we will arrange both the x_1 and x_2 intersection points in some order, either increasing or decreasing order of their magnitude. With these points we have associated surface numbers 1 and 2. This order of their magnitude will give us the order of the surfaces encountered. This information of order of the first encounter or the last encounter by the given line is required in the NXT module of the code DRAGON.

For this we use bubble SORT routine [20]. While using this routine both the x-intersection points are sorted. Here the difference with the normal sorting is that along with this order of x_i number this routine gives its i number, which represents its surface number i. This way we come to know, in which order the two surfaces of the hexagon are encountered. This routine is also used while sorting intersection points with many hexagons, as will be mentioned in chapter 5.

In the NXT module distances are required to be computed w.r.t. point (x_a, y_a). Let us denote the distances from this point, as u_1 and u_2. Difference between the two gives us the track length u in the hexagon. These track distances for both the surfaces, are given by

$$u_1 = \pm\sqrt{(x_1 - x_a)^2 + (y_1 - y_a)^2} \tag{3.6}$$

and

$$u_2 = \pm\sqrt{(x_2 - x_a)^2 + (y_2 - y_a)^2} \tag{3.7}$$

Such computations are done for various lines.

3.2.3 Flow chart for Tracking algorithm in one hexagon

All the computations are summarized in the flow chart Fig. 3.4.

3.2.4 Computation of tracks for various lines

In the Carlvik's [7] method of collision probability, parallel lines are drawn, such that they cover the 2D geometry under consideration. These lines are at one angle. Tracks lengths in the given geometry are computed for all the lines. This process is then repeated for all the angles, specified by the numerical quadrature method selected to compute integrals in the collision probability expressions. The number of tracking directions are given by N and the tracking density is given by T/cm mentioned in chapter 2. The NXT module computes tracks inside cylindrical pins (Figs. 3.5 and 3.6). These tracks are used to compute collision probabilities which are used to compute fluxes by using Eq. (2.12). The results so obtained

are present below.

3.3 Computations for a 2D hexagon cell

In this section, we present the results of one group calculations that have been made to check the accuracy of the algorithm, developed in this thesis, and implemented in the NXT module of the code DRAGON [24]. Cross-sections are taken from [30]. These cross-sections are given for under moderated lattices and are presented in Table 3.1.

A pin or a cluster of pins, depending upon the case is surrounded by moderator region which fills the hexagonal cell. Constant source is assumed in the moderated region. The results so obtained are compared with the results obtained by the EXCELT module. In the case of one hexagon two cases are considered.

a) Hexagon cell with one cylindrical pin

   Here one cylindrical fuel pin is surrounded by moderator in a hexagon cell (Fig. 3.5). The dimensions of the cell are

   - Side of the hexagon = 0.562563 cm
   - Radius of the cylindrical pin = 0.4 cm

b) Hexagon cell with a cluster of six cylindrical pins

   Here six cylindrical fuel pins are placed at equal distance from the center of the hexagon cell. All these pins are surrounded by moderator. The moderator region is subdivided into two regions. One region surrounds all the six cylindrical pins, as shown in Fig. 3.6. Dimensions of the cell are

   - Side of the hexagon = 0.562563 cm
   - Radius of moderator region surrounding six pins = 0.4 cm.
   - Radius of each of the six cylindrical pins of the cluster (of pins) = 0.1 cm.
   - All the six pins are placed at a radius = 0.3 cm.

3.3.1 Plots of tracks obtained

For the case of a hexagonal cell with one pin, the tracks plotted by the NXT module are shown in Figs. 3.7. The selected track directions N and track density T/cm in this case are 3 and 10 respectively. In the case of a cluster of six pins, the plots are shown in 3.8. In this case the selected track directions N and track density T/cm are 8 and 10 respectively. These values are selected such that tracks are distinctively visible.

Table 3.1 Cross-sections for fuel and moderator

Cross-section	Water	Fuel
Σ	1.0770816	0.677993
Σ_s	1.0767	0.44228

3.3.2 Comparison of results for 2D hexagon cells

Average neutron fluxes have been obtained by using both the modules EXCELT and NXT for various values of N and T, mentioned in chapter 2. These are presented, for the case of one pin 2D hexagon cell in Tables 3.2 and 3.3. For the case of a cluster of pin cell, the results are presented in Tables 3.4 and 3.5. These results are compared in Tables 3.6 and 3.7.

In the NXT module volumes V and surfaces S are computed numerically, this is described in chapter 2. These numerically computed values are compared with the exactly computed values. The percentage errors V on volumes and S on surfaces are given in Tables 3.3 and 3.5. For the values of $N = 16$ and $T = 50.0/cm$, there is a percentage error of 0.03 % on volumes and on surfaces S is 0.1%, in both the cases of single pin cell as well as for the case of cluster of pins cell (Tables 3.3 and 3.5).

In the case of a single pin cell both the NXT and EXCELT results are found to converge and match (Table 3.6).

When there is a cluster of pins in the hexagon cell, there is a convergence problem in the EXCELT module after $N = 10$ and $T = 8.0/cm$ (Table 3.4). The results of both the modules EXCELT and NXT are found to match very well for fuel region 1. This can be seen from Tables 3.7. In the moderator region, the results differ by 0.186 in region 2 and by about 0.1 in region 3.

Table 3.2 Computation of average fluxes in 2D hexagon cell with one pin (EXCELT)

N	T	FLUX 1	FLUX 2
4	2.0	2.694	2.739
4	4.0	2.694	2.847
4	8.0	2.694	3.026
8	8.0	2.694	3.021
10	8.0	2.694	3.020
10	10.0	2.694	2.880
10	16.0	2.694	2.853
16	16.0	2.694	2.854
16	50.0	2.694	2.907

Table 3.3 Computation of average fluxes in one pin 2D hexagon cell (NXT)

N	T	FLUX 1	FLUX 2	V	S
2	2.0	2.694	2.760	-1.4	11.1
4	2.0	2.694	2.760	-1.5	-1.5
4	4.0	2.694	3.012	-4.1	-5.7
4	8.0	2.694	2.859	-0.1	-2.5
8	8.0	2.694	2.857	0.1	-2.5
10	16.0	2.694	2.932	-0.2	0.2
16	16.0	2.694	2.932	-0.3	-0.1
16	20.0	2.694	2.903	-0.2	0.1
16	50.0	2.694	2.913	0.03	0.1

Table 3.4 Computation of fluxes in 2D hexagon cell with a cluster of six pins (EXCELT)

N	T	FLUX 1	FLUX 2	FLUX 3
4	2.0	1.4184E+01	1.4503E+01	1.4518E+01
4	4.0	1.4184E+01	1.4503E+01	1.4518E+01
4	8.0	1.4184E+01	1.4476E+01	1.4602E+01
8	8.0	1.4184E+01	1.4473E+01	1.4561E+01
10	8.0	1.4185E+01	1.4275E+01	1.4643E+01
10	10.0	* PIJRHL:	PROBLEM OF	ACCELERATION

Table 3.5 Computation of fluxes in 2D hexagon cell with a cluster of six pins (NXT)

N	T	FLUX 1	FLUX 2	FLUX3	V	S
2	2.0	1.4185E+01	1.4507E+01	1.4382E+01	-1.4	11.1
4	2.0	1.4185E+01	1.4391E+01	1.4310E+01	-1.5	11.1
4	4.0	1.4184E+01	1.4464E+01	1.4643E+01	-4.2	-4.1
4	8.0	1.4184E+01	1.4546E+01	1.4550E+01	-0.1	-2.5
8	8.0	1.4184E+01	1.4525E+01	1.4536E+01	0.1	-2.5
10	16.0	1.4184E+01	1.4460E+01	1.4553E+01	-0.2	0.1
16	16.0	1.4184E+01	1.4460E+01	1.4554E+01	-0.2	0.1
16	50.0	1.4185E+01	1.4461E+01	1.4548E+01	0.03	0.1

Table 3.6 Comparison of average neutron fluxes for one pin 2D hexagon cell

Module	FLUX 1	FLUX 2
EXCELT(2D)	2.694	2.907
NXT(2D)	2.694	2.913

Table 3.7 Comparison of average fluxes in 2D hexagon cell with a cluster of six pins

Module	FLUX 1	FLUX 2	FLUX3
EXCELT(2D)	1.4185E+01	1.4275E+01	1.4643E+01
NXT(2D)	1.4185E+01	1.4461E+01	1.4548E+01

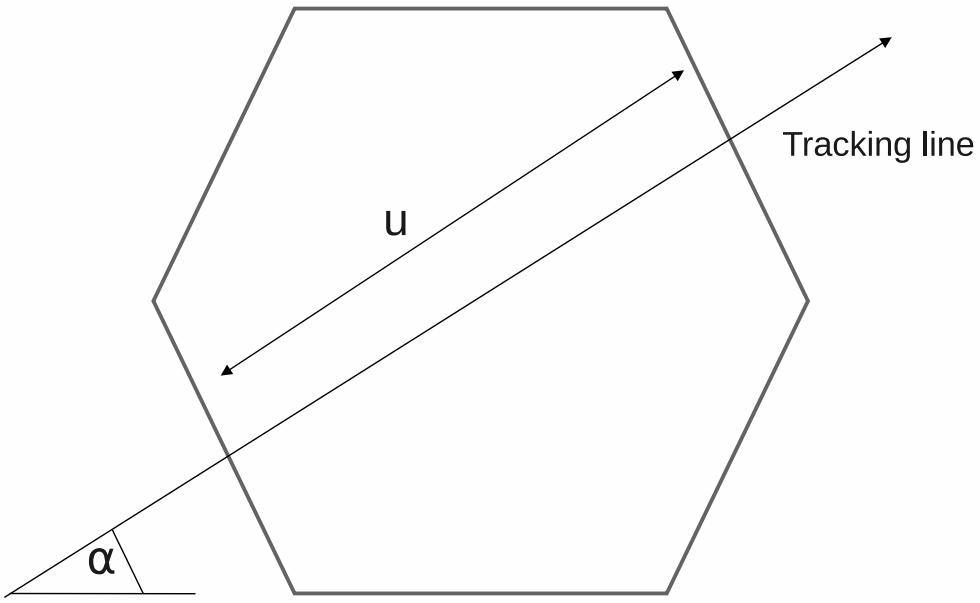


Figure 3.1 2D Track in one Hexagon

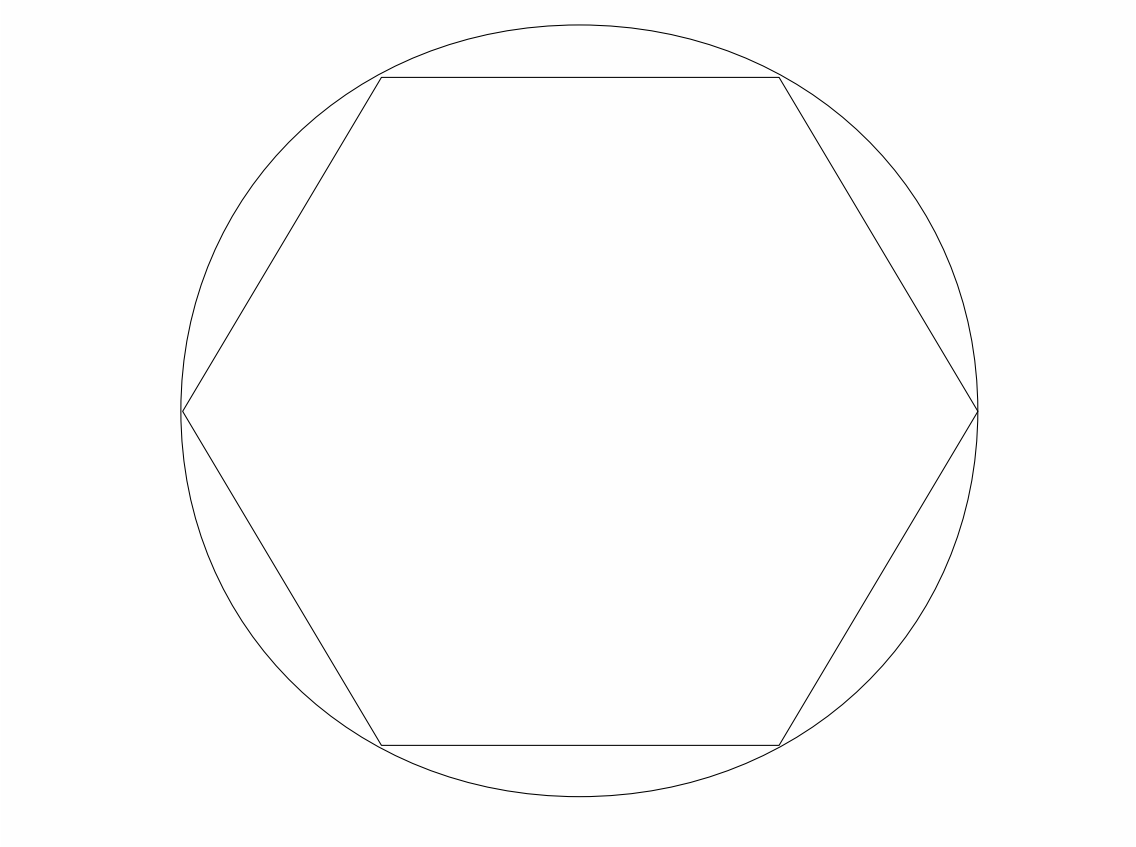


Figure 3.2 Circle surrounding one Hexagon

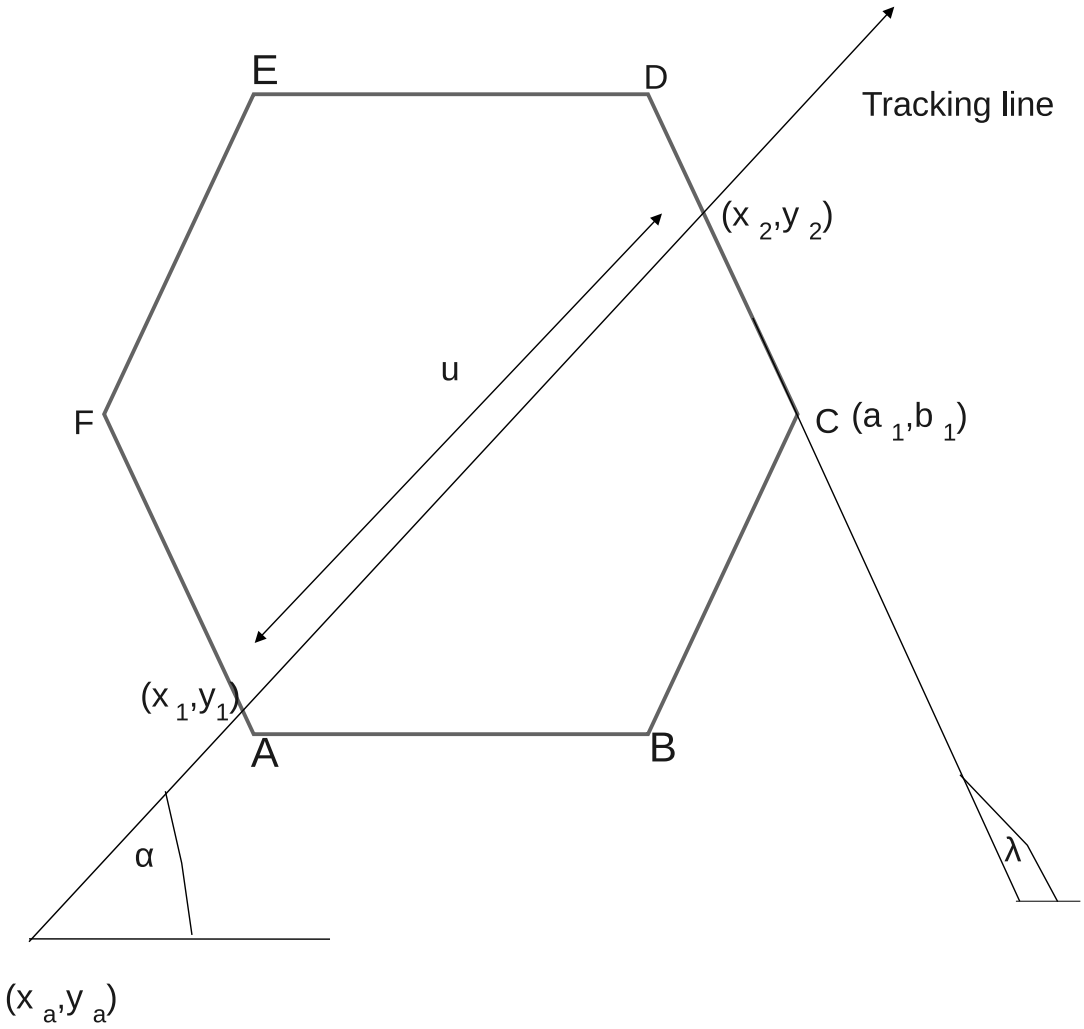


Figure 3.3 Notations for track computations in one 2D hexagon

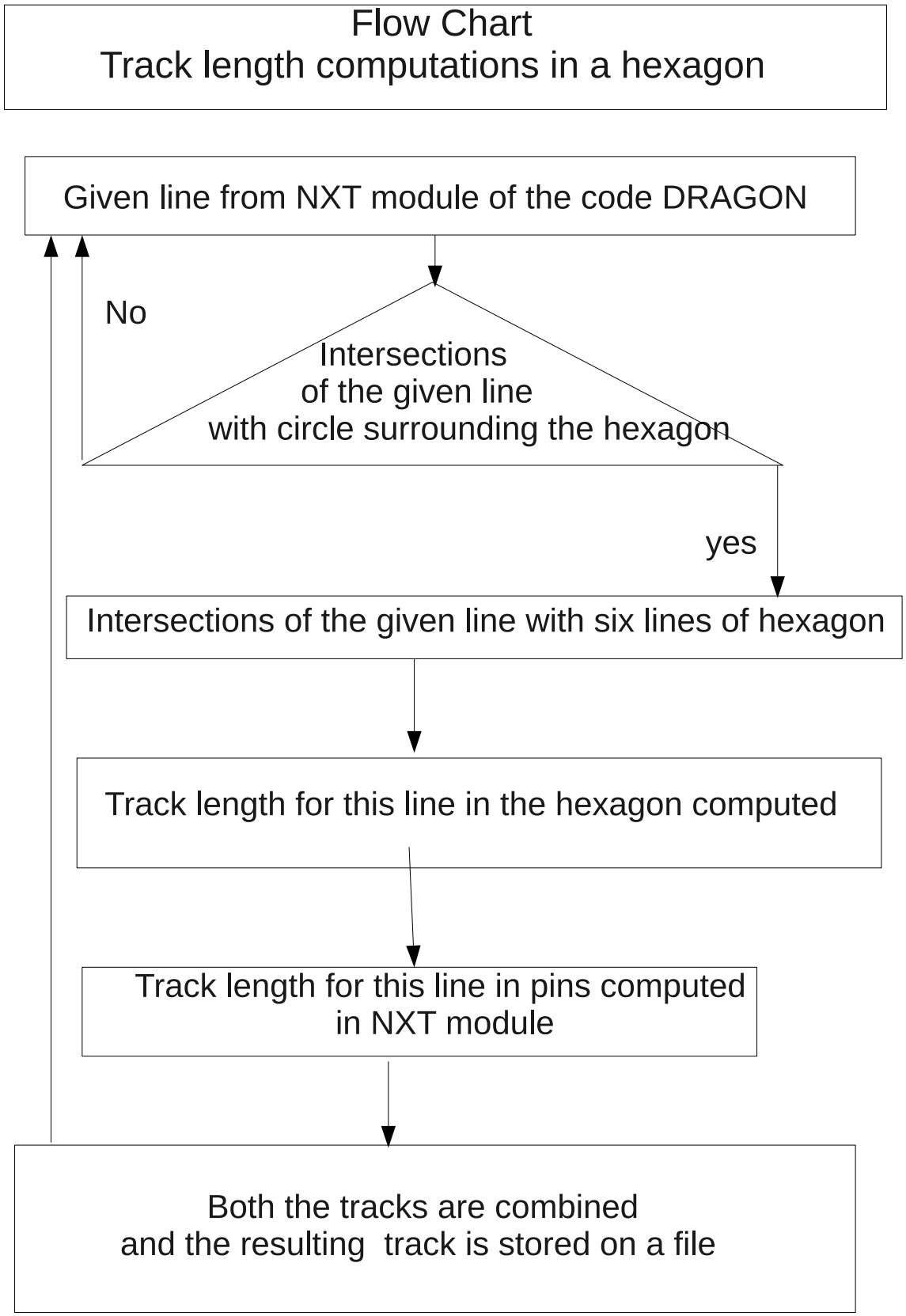


Figure 3.4 Flow chart for flux computations in a 2D hexagon cell

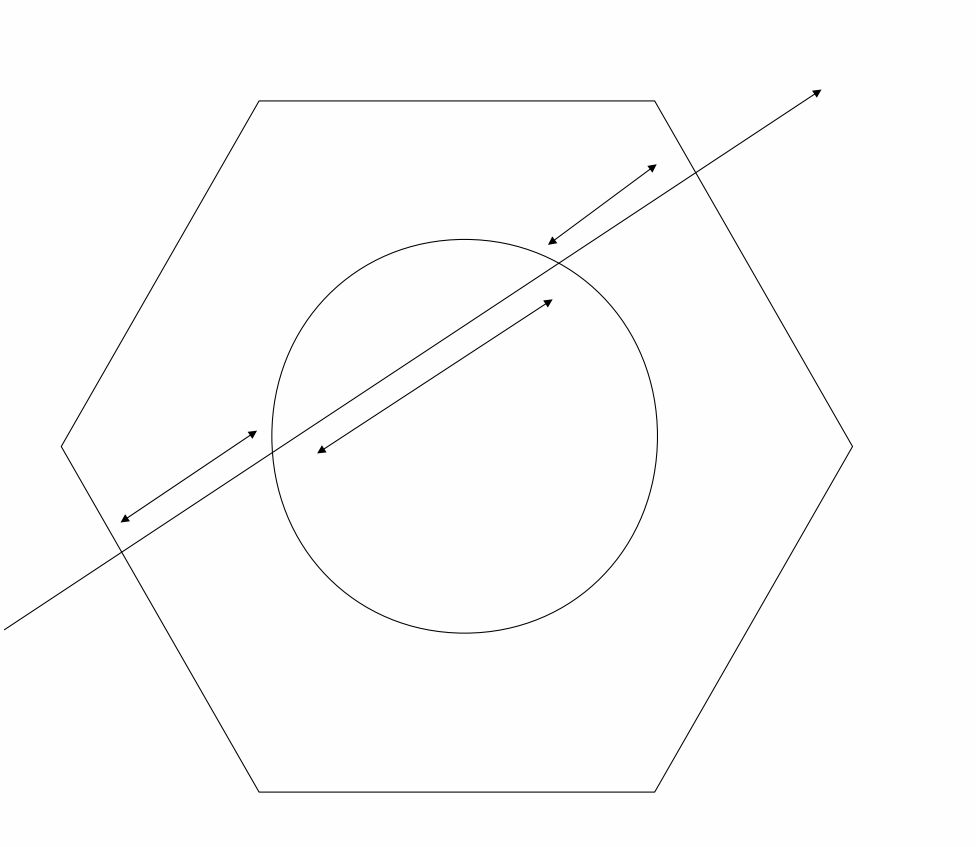


Figure 3.5 Track lengths in one hexagon cell

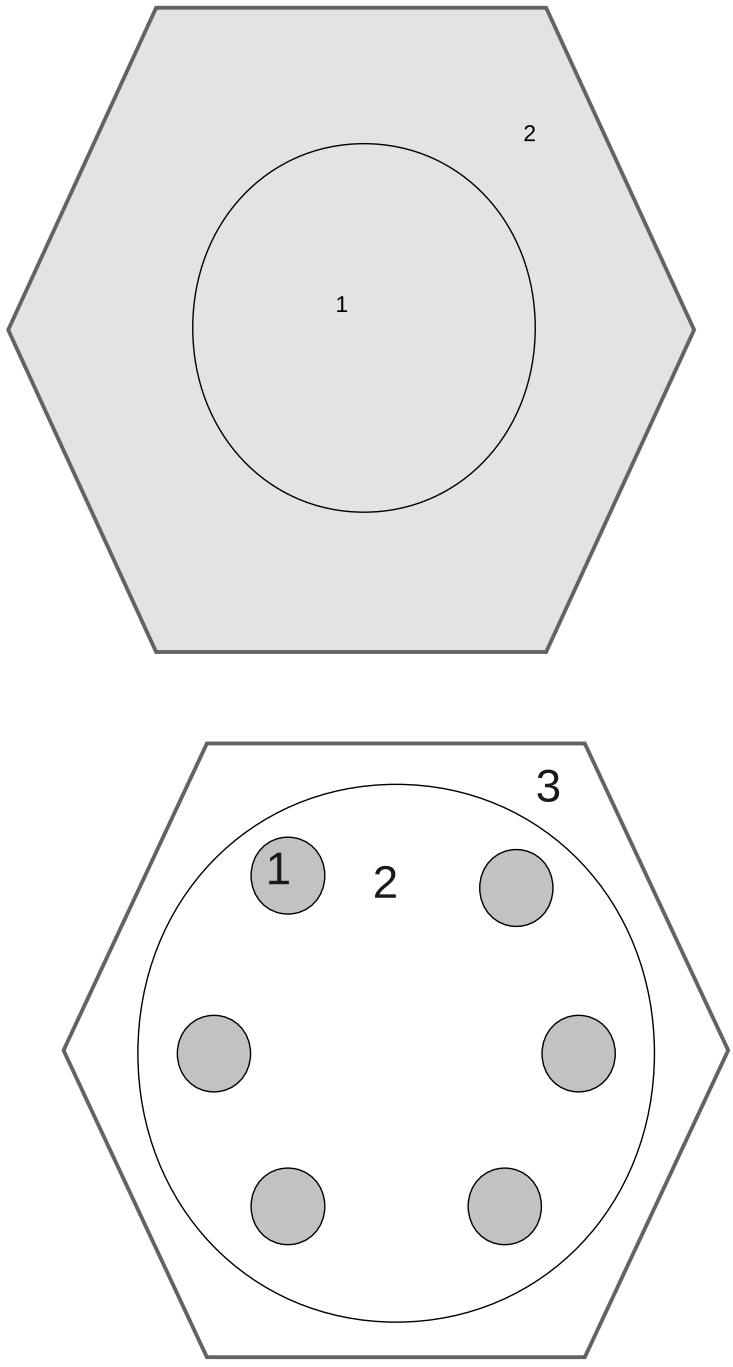


Figure 3.6 Region numbers in one pin hexagon cells (2D)

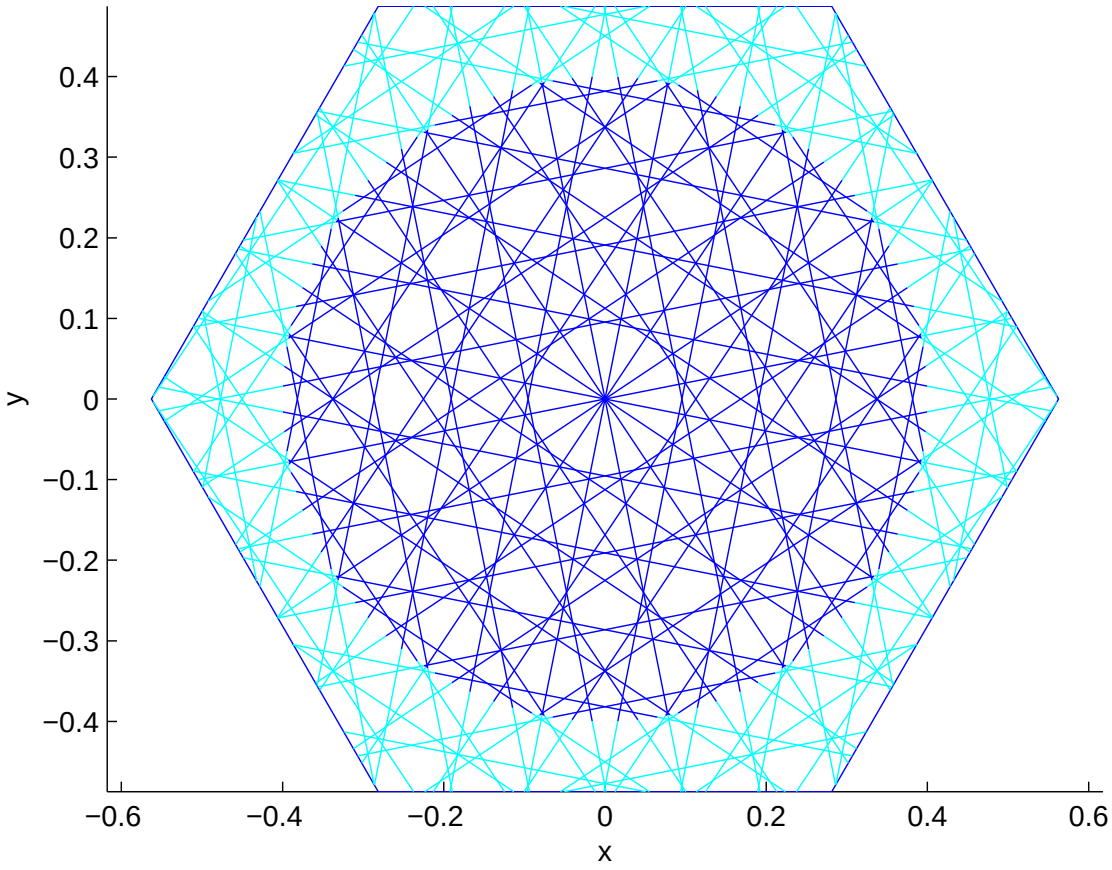


Figure 3.7 Tracks obtained in one pin 2D hexgon cell

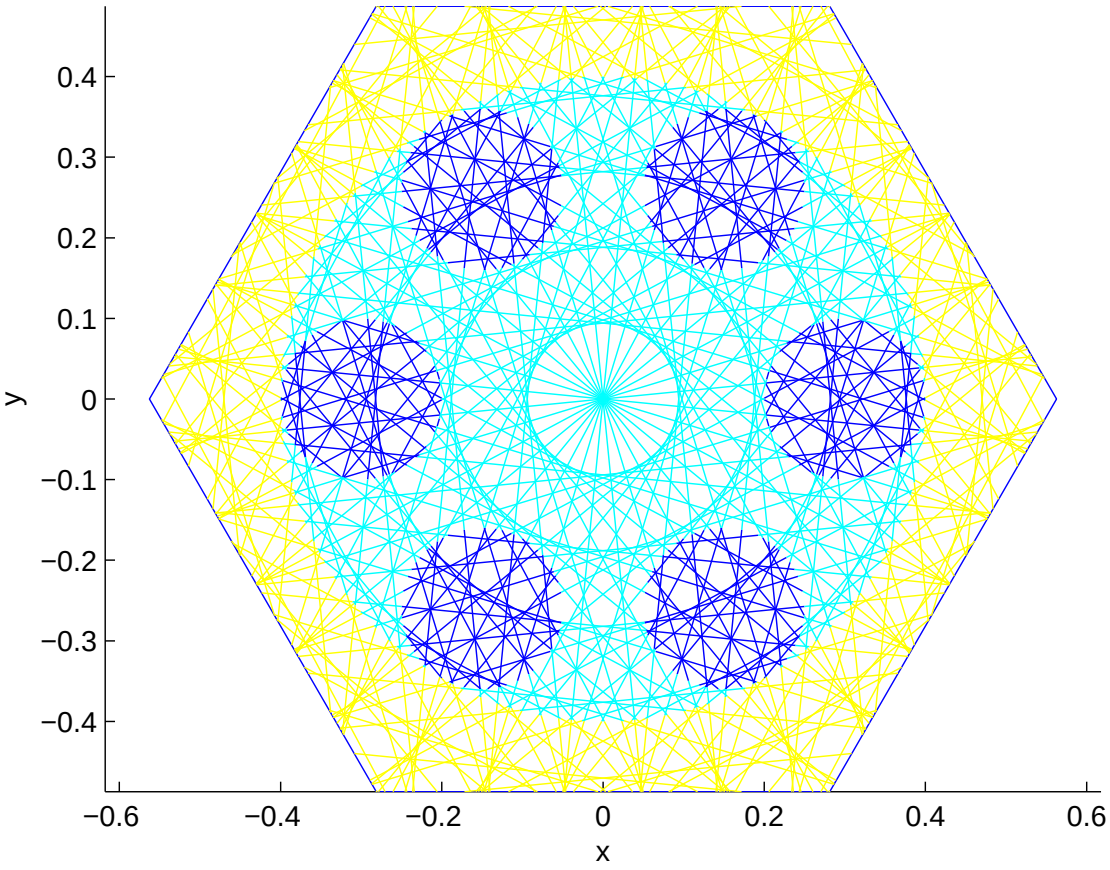


Figure 3.8 Tracks obtained in a 2D six pin hexagon cell

CHAPTER 4

TRACKING ALGORITHM AND COMPUTATION OF FLUXES IN A 3D HEXAGON CELL

4.1 Introduction

In this Chapter we will describe the method for computing tracks in a three dimensional (3D) hexagonal geometry. These tracks will be combined with the tracks in a cylindrical pin or a cluster of pins (depending on the case). We will present the results obtained by this approach.

4.2 Tracking algorithm in three dimensions

In this section we describe our algorithm to compute tracks in one 3D hexagon. This algorithm is also used to compute tracks in the case of multi (3D) hexagonal assemblies. This will be presented in Chapter 5.

4.2.1 Tracking algorithm for one hexagon

In the method of collision probability [7], parallel lines are drawn, such that they cover the 3D geometry under consideration. Each line has a specific direction and a starting point. After evaluating the intersection points of these lines with the hexagon faces tracks lengths are computed. This process is repeated for various directions and intitial positions in space, depending on the geometry.

4.2.2 Checking before computations

Let us consider that a line makes α angle with the x-axis and β angle with the vertical z-axis. In addition let us assume that the line passes through the hexagon as shown in Fig. 4.1. For computing tracks in one hexagon, for this line, we are required to compute intersection points of the given line with the sides of the hexagon as can be seen from this figure. This line intersects the hexagon on two sides at respective heights of z_1 and z_2. On the $(x-y)$ plane (2D plane) the projections of this line are u_1 and u_2 respectively. These u_1 and u_2 are at an angle of α with the x-axis and are computed as mentioned in chapter 3. Before computing any 3D track, first we will check if the given line intersects a circle, which surrounds this hexagon, in 2D plane. If the line intersects this circle in 2D plane, then only it may intersect

the hexagon in 3D plane. Equations for checking of intersection points of a line with a circle are given in Appendix A. In this chapter we will develop equations such that by using these u_1, u_2 and angle β values, we compute 3D track length inside the hexagon. Before that we will describe the direction cosines used in the code DRAGON.

4.2.3 Direction cosines in 3D

The direction cosines used, in 3D, in the code DRAGON are

$$dirtrk(1) = \sin\beta\cos\alpha \tag{4.1}$$

$$dirtrk(2) = \sin\beta\sin\alpha \tag{4.2}$$

$$dirtrk(3) = \cos\beta \tag{4.3}$$

For 3D geometries angle α varies from 0 to 2π and angle β varies from 0 to $\pi/2$ as shown in Fig. 4.2. The direction of a line depends on the magnitude and sign of these direction cosines. For these β values (0 to $\pi/2$) both the values of $\sin\beta$ and $\cos\beta$ are positive. The sign of these direction cosines depends on the α values only. Middle of the α angle in the four quadrants are respectively $\pi/4$, $3\pi/4$, $5\pi/4$ and $7\pi/4$. We number these quadrants as $1, 3, 5, 7$ respectively. We will describe how to compute track for one line. This process is repeated for all the lines.

4.2.4 Computations for rectangular surfaces

For computing the tracks in a 3D hexagon, we will divide the hexagon into various parallel $(u - z)$ planes as shown in Figs. 4.3 and Fig. 4.4. These planes are selected such that the 2D tracks u lie on these planes. These planes are perpendicular to the base of the hexagon. The reason for doing this is that once we know the 2D tracks, 3D tracks will lie on these planes. This plane has four boundaries top H_t, bottom H_b and two side surfaces L and R. Side surfaces represent the intersection of the given line with the side surfaces of the hexagon. A line may cut any of the four surfaces as shown in Fig. 4.5. We need to compute tracks for a rectangular geometry only. This process will be repeated for all the planes. In this way we will be able to compute tracks for all the lines and for all the angles, in a 3D hexagon, by using the equation mentioned below.

4.2.5 Equations for tracking algorithm in 3D

Equations are developed to compute tracks on each plane. These planes have rectangular boundaries. In this section we will describe the tracking algorithm developed for computing

tracks for a rectangular boundary. For doing this, equations are developed such that one has to deal with 2D equations at one time. Initially equations are developed in 2D (x-y) plane and we get the resulting track u (Fig. 4.6a). Then the equations are developed in the same way for 2D $(u - z)$ plane and we get the resulting track d_3 (Fig. 4.6b). Track d_3 represents the track in 3D, it can be expressed as

$$d_3 = z \sec \beta \tag{4.4}$$

From this equation we observe that we are required to compute z, for computing d_3 the 3D track distance. For computing z values, we will develop equations. For completeness, we will write equations of Chapter 3 again.

4.2.6 Equation in 2D $(x - y)$ plane

Equation of a line which makes an angle α with the x-axis (Fig. 4.6a) is

$$\frac{y - y_0}{x} = \tan \alpha \tag{4.5}$$

or

$$y = x \tan \alpha + y_0 \tag{4.6}$$

when $x = 0$

$$y = y_0 \tag{4.7}$$

Line passes through the point (x_a, y_a, z_a). One gets

$$y_0 = y_a - x_a \tan \alpha \tag{4.8}$$

The resulting track distance u in 2D $(x - y)$ plane is

$$u = \pm\sqrt{(x^2 + y^2)} \tag{4.9}$$

In the NXT module the distances are required to be computed w.r.t. the point (x_a, y_a, z_a). In this case u is given by

$$u = \pm\sqrt{(x - x_a)^2 + (y - y_a)^2} \tag{4.10}$$

Here u is negative when $(x_a, y_a) < 0$.

It is mentioned in 2D $(x-y)$ approach (chapter 3), that the (x, y) intersection points with

both the sides of one plane are denoted as (x_1, y_1) and (x_2, y_2). We get by using Eq. (4.10).

$$u_1 = \pm\sqrt{(x_1 - x_a)^2 + (y_1 - y_a)^2} \tag{4.11}$$

$$u_2 = \pm\sqrt{(x_2 - x_a)^2 + (y_2 - y_a)^2} \tag{4.12}$$

The track distances u_1 and u_2 are computed w.r.t. the point (x_a, y_a). Their relative position on the 2D (x, y) plane can be obtained by comparing the respective points x_i (i=1,2) with the point x_a. If both of these x_i (i=1,2) points lie on one side of x_a, the required track length in the body is difference of the absolute values of u_1 and u_2. If both of these points x_i (i=1,2) lie on two sides of the point x_a, the required track length in the body is sum of the absolute values of u_1 and u_2. To consider this fact, signs of u_1 and u_2 are considered accordingly.

All the lines are facing upwards in the upper quadrant as mentioned in chapter 2. We consider distances below the point (x_a, y_a, z_a) as negative. For lines facing the quadrants 1 and 7 (Fig. 4.2) we define

- when $x_1 < x_a$, then u_1 is negative.
- when $x_2 < x_a$, then u_2 is negative.

For lines facing the quadrants 3 and 5 we define

- when $x_1 \geq x_a$, then u_1 is negative.
- when $x_2 \geq x_a$, then u_2 is negative.

These values of u are used to compute z, which is required to compute d_3 the track distance in 3D plane. For this purpose, the following equations are developed.

4.2.7 Equations in $(u - z)$ plane (2D)

Equation of a line which makes an angle β with the z-axis, as can be seen from Fig. 4.6b, is

$$\frac{z - z_0}{u} = \cot\beta \tag{4.13}$$

Equation for z and u can be expressed as

$$z = z_0 + u\cot\beta \tag{4.14}$$

Line passes through $(0, 0, 0)$.

$$z_0 = 0 \tag{4.15}$$

This is a trivial solution.

Since line passes through (x_a, y_a, z_a) also, we have $u = 0$ at (x_a, y_a) (Eq. 4.10). We get

$$z_0 = z_a \tag{4.16}$$

4.3 Computation of 3D track distances

For all the side surfaces we compute track distances from the point (x_a, y_a, z_a). Depending upon the first or second intersection of a line with any of the four boundaries of a $(u - z)$ plane, 3D track distance is computed. Let the first encounter be with '1' surface and second encounter be with '2' surface and the corresponding vertical distances are z_1' and z_2'. These distances are from the origin $(0, 0, 0)$. Let us define distances z_1 and z_2, the distances from the point (x_a, y_a, z_a). We get by using Eqs. (4.14) and (4.16).

$$z_1 = z_1' - z_a = u_1 \cot\beta \tag{4.17}$$

and

$$z_2 = z_2' - z_a = u_2 \cot\beta \tag{4.18}$$

These values of z_1 and z_2 are used in computing track distances d_{31} and d_{32}, by using Eq. 4.4. First intersection distance, of the given line from the point (x_a, y_a, z_a), is denoted as d_{31}. Second intersection distance, of the given line from the point (x_a, y_a, z_a), is denoted as d_{32}. Difference of these two distances gives us the required track d_3, inside the 3D hexagon.

Track length computations for side surfaces, top and bottom surfaces are done in the following way.

4.3.1 Computations for side surfaces

When the track distances in the $(x-y)$ plane are u_1 and u_2 (Eqs.4.11 and 4.12), the equations in the $(u - z)$ plane are obtained by using Eqs. (4.17) and (4.18), we get z_1 and z_2, the respective intersection lengths of the given line for both the side surfaces from the point (x_a, y_a, z_a).

The first surface and the second surface encountered, by the line, from the point (x_a, y_a, z_a) (Eqs. (4.17) and (4.18)) are computed as

$$d_{31} = z_1 \sec\beta \tag{4.19}$$

or

$$d_{31} = (z_1' - z_a) \sec\beta \tag{4.20}$$

and

$$d_{32} = z_2 \sec\beta \tag{4.21}$$

or

$$d_{32} = (z_2' - z_a) \sec\beta \tag{4.22}$$

4.3.2 Computations for top and bottom surfaces

Since all lines are facing upward direction, bottom surface is encountered first and top surface later. Using Eqs. (4.20) and (4.22), where z_1' and z_2' are replaced by H_b and H_t respectively, we get

$$d_{31} = (H_b - z_a) \sec\beta \tag{4.23}$$

and

$$d_{32} = (H_t - z_a) \sec\beta \tag{4.24}$$

The order of the side encountered by the given line depends on its direction. Depending upon the direction of α and β, tracks are computed in various cases. For different directions, relations are mentioned in Appendix B.

4.3.3 Computation of collision probabilities and fluxes

For computing the collision probability integrals, one is required to evaluate track lengths for various lines at different angles α, β and for various x and y. These lines are generated in the NXT module of DRAGON, depending on the numerical method of approximating these integrals. For all the four integrals we use equal weight quadrature method [6] described in Chapter 2.

These probabilities are used to compute fluxes by using Eq.(2.12). The results so obtained are present below.

4.4 Computations for a 3D hexagon cell

In this section, we present the results of one group calculations that have been made to check the accuracy of the algorithm, developed in this thesis, and implemented in the NXT module of the code DRAGON [24]. Cross-sections are taken from [30, 33]. These cross-sections are given for under moderated lattices and are presented in Table 4.1.

A pin or a cluster of pins, depending upon the case is surrounded by moderator region which fills the hexagonal cell (Fig. 4.7). Constant source is assumed in the moderated region. The results so obtained are compared with the results obtained by the module EXCELT. In the case of one 3D hexagon cell, two cases are considered.

a) One 3D hexagon cell with one cylindrical pin

Here one cylindrical fuel pin is surrounded by moderator in a hexagon cell. The dimensions of the cell are

- Side of the hexagon = 0.562563 cm
- Height of the hexagon = 1.0 cm (-0.5 cm to +.5 cm)
- Radius of the cylindrical pin = 0.4 cm

b) One 3D hexagon cell with a cluster of six cylindrical pins

Here six cylindrical fuel pins are placed at equal distances from the center of the hexagon cell. All these pins are surrounded by moderator. The moderator region is divided into two regions. One region surrounds all the six cylindrical pins, as shown in Fig. 4.7. The dimensions of the cell are

- Side of the hexagon = 0.562563 cm
- Height of the hexagon h = 1.0 cm (-0.5 cm to +.5 cm)
- Radius of moderator region surrounding six pins = 0.4 cm.
- Radius of each of six cylindrical pins of the cluster (of pins) = 0.1 cm.
- Six pins are placed uniformly at a radius = 0.3 cm.

Table 4.1 Cross-sections for fuel and moderator

Cross-section	Water	Fuel
Σ	1.0770816	0.677993
Σ_s	1.0767	0.44228

4.4.1 Plots of tracks obtained in one 3D hexagon cell

Plots inside a hexagon cell with one pin are shown in Figs. 4.8. The selected track directions N and track density T/cm^2 in this case are 4 and 2 respectively.

Plots in the case of a hexagon cell with a cluster of six pins are shown in Fig. 4.9. In this case the selected track directions N and the track density T/cm^2 are 8 and 8 respectively.

These values are selected such that tracks are distinctively visible.

4.4.2 Comparison of results for 3D hexagon cells

Results have been obtained, for various values of N and T defined in chapter 2, by using the NXT module. For EXCELT module the results are presented for various values of $N1$, $N2$, $T1$ and $T2$ defined in chapter 2. These results are presented, for the case of one pin 3D hexagon cell in Tables 4.2 and 4.3. For the case of a cluster of six pins cell, the results are presented by using NXT module in Table 4.4. The EXCELT module does not consider 3D clusters. For this reason reflective boundary conditions are applied at the top and bottom surfaces of the hexagonal cells and results are compared with 2D EXCELT results (Tables 4.5 and 4.6).

The NXT module gives percentage errors on volume V and on surfaces S mentioned in chapter 2. For values of $N = 16$ and $T = 50.0/cm^2$, there is a percentage error of 0.5 % on volumes and on surfaces it is 1.6%, in both the cases of a single pin cell as well as for a cluster of pins cell (Tables 4.3 and 4.4).

In the case of a single pin 3D cell both the NXT and EXCELT results match for fuel region 1. There is a maximum difference in results by 0.177, in the moderator region 2 (Table 4.5).

When there is a cluster of pins in the hexagon cell, the results in the case of NXT(3D) and NXT(2D) differ by 0.04 for region 2 and they differ by 0.02 for region 3 (Table 4.6). The results of NXT(3D) differ by 0.2 for region 2 and by 0.08 for region 3 when compared with EXCELT (2D) results.

From the results of EXCELT (Table 4.2), it apprears that convergence is not achieved properly in 3D case even with the tracking density of $T1 = 50.0/cm^2$ and $T2 = 50.0/cm^2$. Similar trend is seen in the case of results of NXT, where the error on surfaces appears to increase in 3D cases. The reason may be that in these cases, that some of the lines are having α tending to $\pi/2$. The numerical computation of $\tan\alpha$ in 2D case might not be computed properly. When 3D tracks are computed by using these values, the error will get amplified, since 3D tracks are larger in value than 2D tracks.

Table 4.2 Average neutron fluxes in a 3D one pin hexagon cell (EXCELT)

N1	N2	T1	T2	FLUX 1	FLUX 2
4	4	4.0	4.0	2.695	2.668
4	4	8.0	8.0	2.695	2.686
8	8	8.0	8.0	2.695	2.685
10	10	8.0	8.0	2.695	2.687
10	10	10.0	10.0	2.695	2.710
10	10	16.0	16.0	2.695	2.707
16	16	16.0	16.0	2.695	2.710
16	16	50.0	50.0	2.695	2.726

Table 4.3 Average neutron fluxes in a 3D one pin hexagon cell (NXT)

N	T	FLUX 1	FLUX 2	V	S
4	2.0	2.693	3.694	26.8	-14.6
4	4.0	2.694	2.902	-5.8	1.2
4	8.0	2.694	2.892	2.1	-2.7
8	8.0	2.694	2.883	4.6	-6.0
10	16.0	2.694	2.986	-3.0	0.9
16	16.0	2.694	3.009	-2.9	0.5
16	50.0	2.694	2.903	0.5	-1.7

Table 4.4 Average neutron fluxes in a 3D hexagon cell (with cluster of 6 pins) (NXT)

N	T	FLUX 1	FLUX 2	FLUX3	V	S
4	4.0	1.4185E+01	1.4421E+01	1.4482E+01	-5.8	1.2
4	8.0	1.4185E+01	1.4453E+01	1.4520E+01	2.1	-2.7
8	8.0	1.4185E+01	1.4447E+01	1.4470E+01	4.6	-6.6
10	16.0	1.4184E+01	1.4442E+01	1.4616E+01	2.9	0.9
16	16.0	1.4184E+01	1.4466E+01	1.4648E+01	-2.8	0.4
16	50.0	1.4184E+01	1.4505E+01	1.4567E+01	0.5	-1.6

Table 4.5 Comparison of average neutron fluxes for one pin 3D hexagon cell

Module	FLUX 1	FLUX 2
EXCELT(2D)	2.694	2.907
EXCELT(3D)	2.695	2.726
NXT(2D)	2.694	2.913
NXT(3D)	2.694	2.903

Table 4.6 Comparison of average neutron fluxes for 3D hexagon cell (with cluster of 6 pins)

Module	FLUX 1	FLUX 2	FLUX3
EXCELT(2D)	1.4185E+01	1.4275E+01	1.4643E+01
EXCELT(3D)	-	-	-
NXT(2D)	1.4185E+01	1.4461E+01	1.4548E+01
NXT(3D)	1.4184E+01	1.4505E+01	1.4567E+01

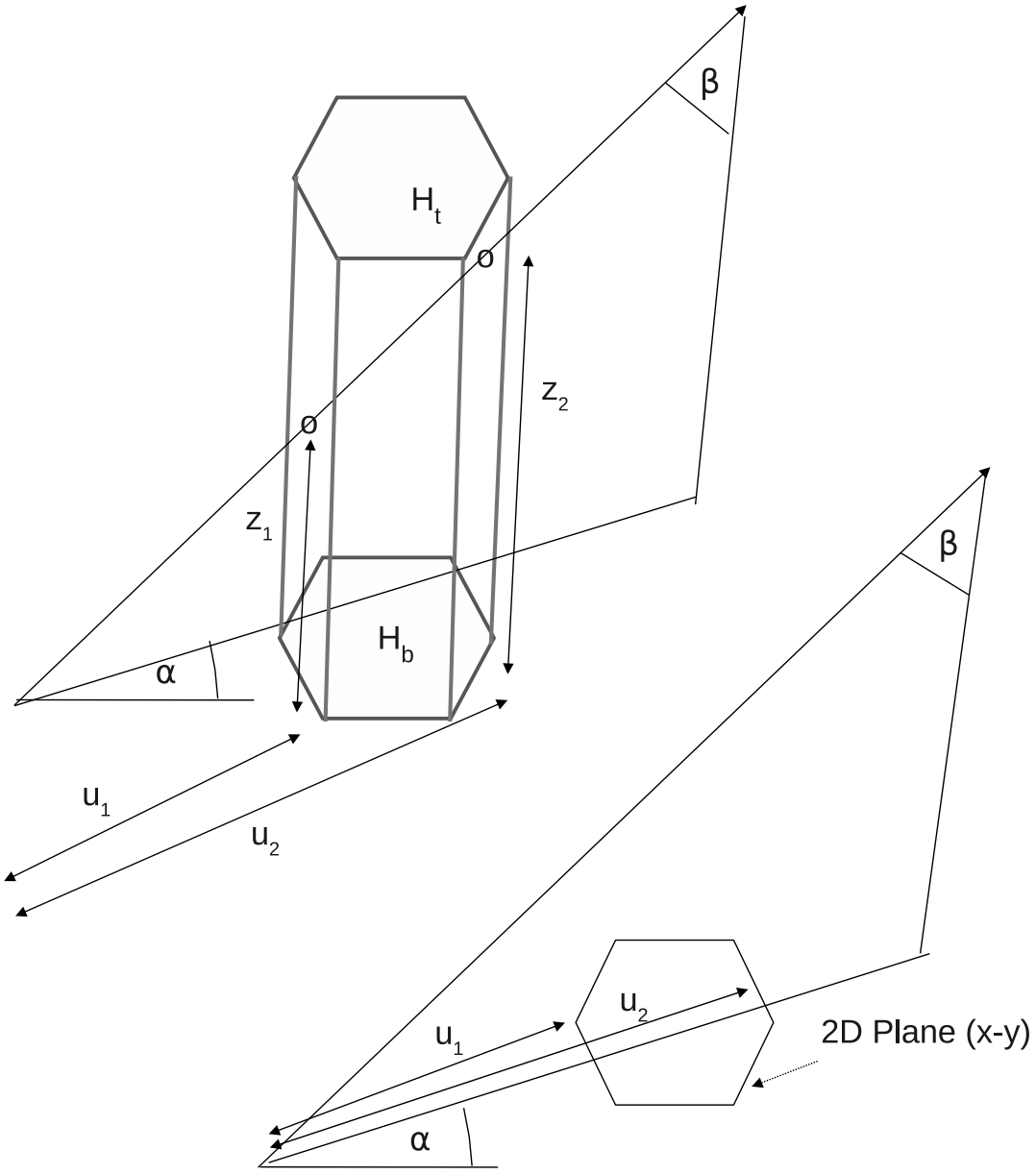


Figure 4.1 Interaction of a line with a 3D hexagon and 2D projections

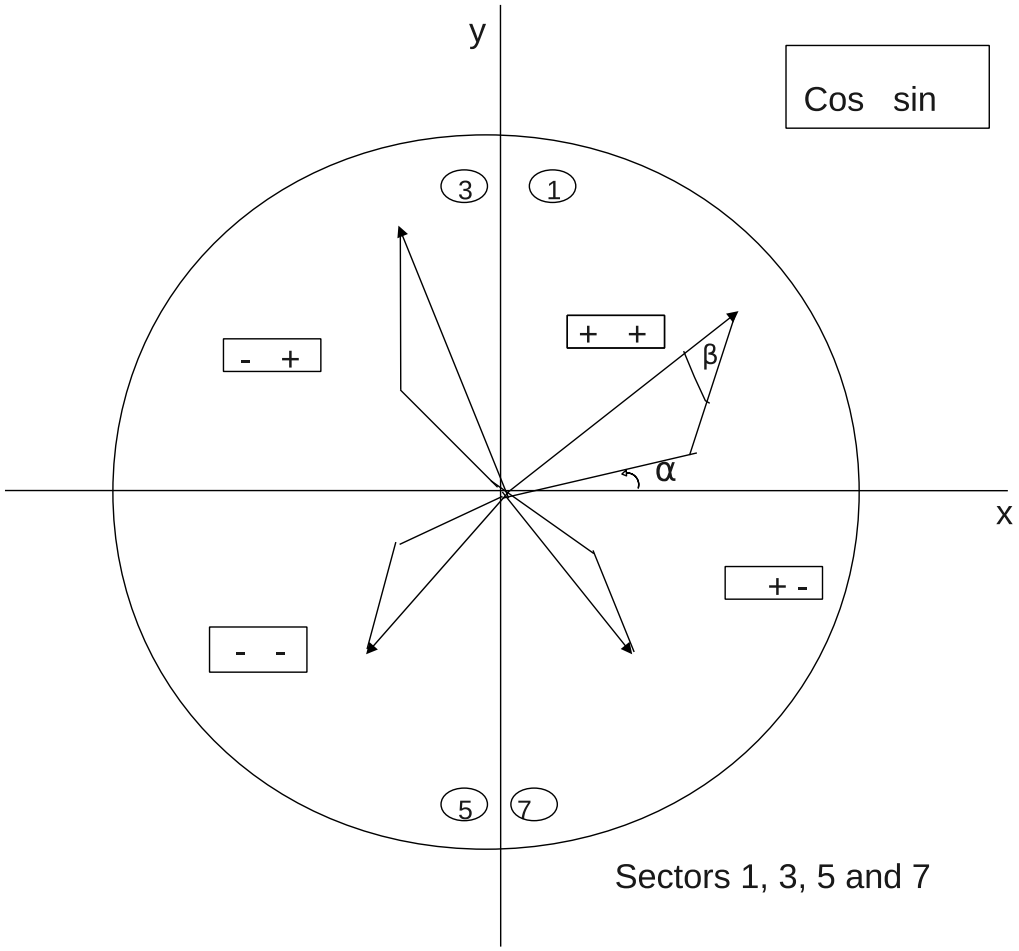


Figure 4.2 Direction of lines and values of $\cos\alpha$ and $\sin\alpha$

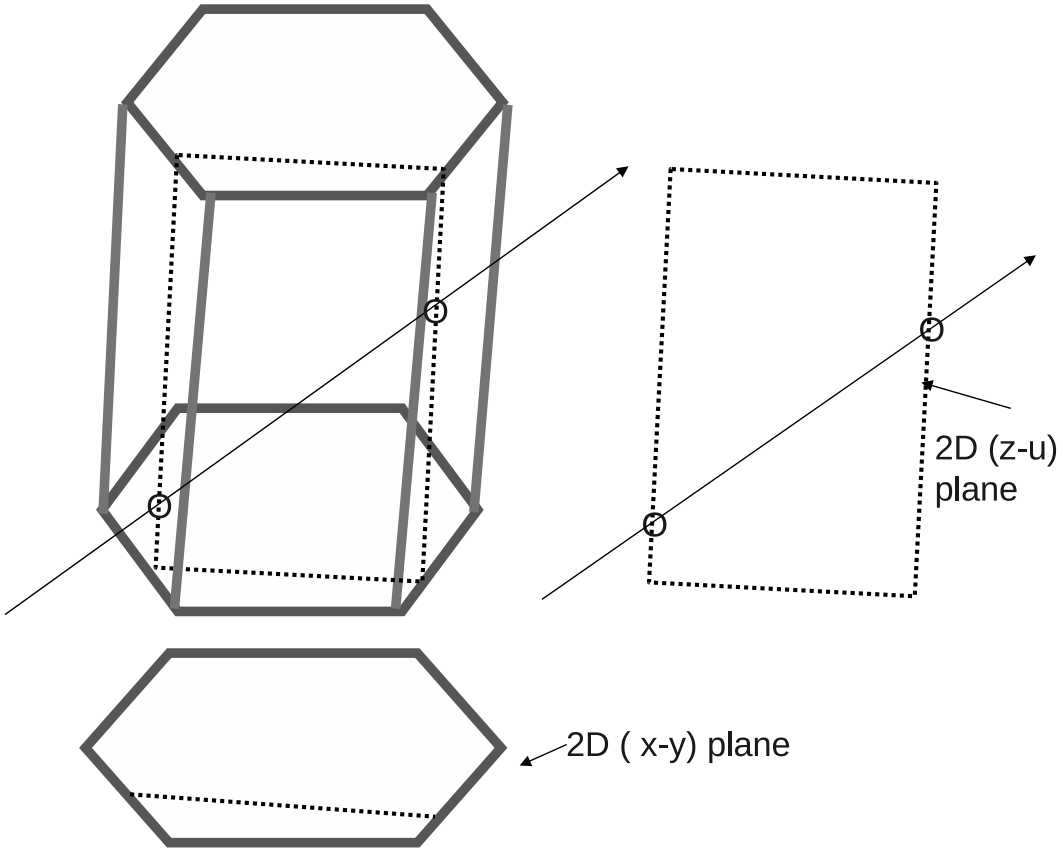


Figure 4.3 Intersection of a line with a 3D hexagon

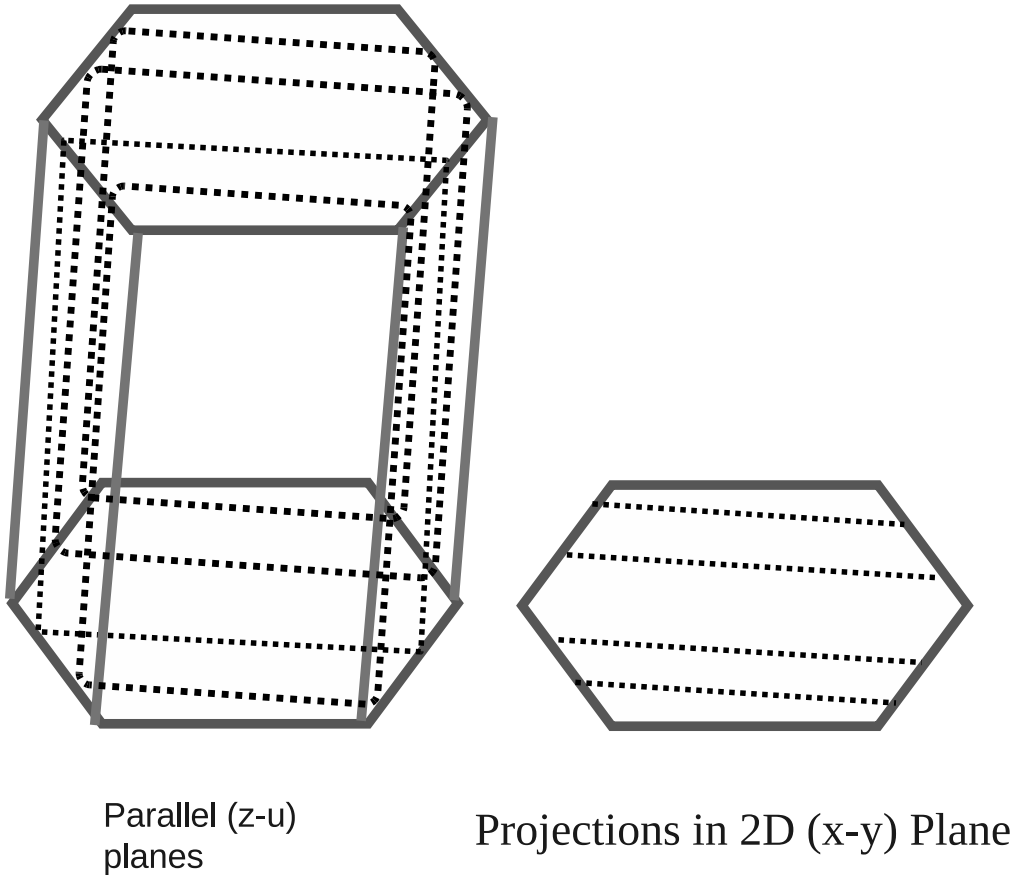


Figure 4.4 3D Hexagon and various parallel planes

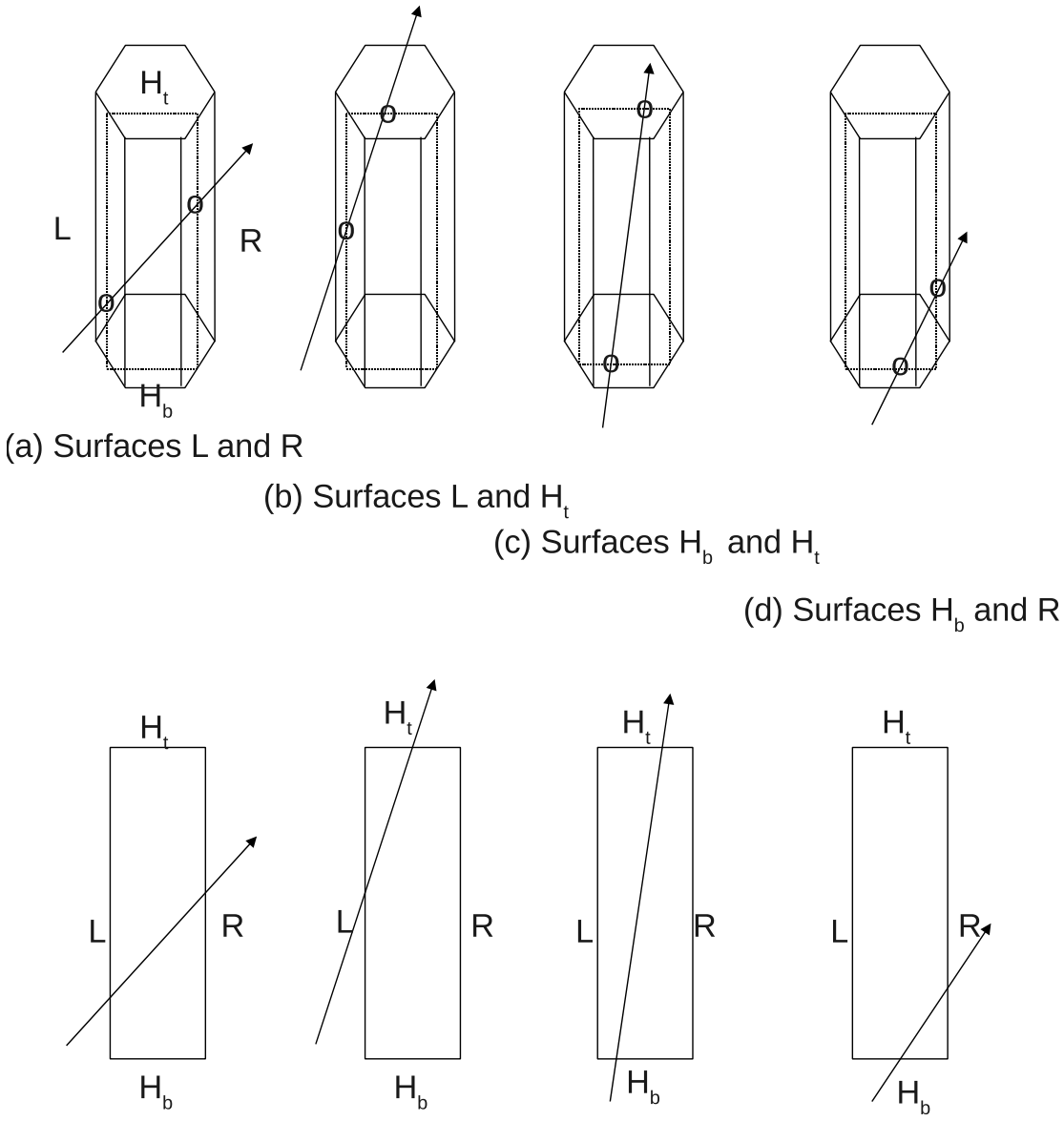


Figure 4.5 Intersection of lines with 3D hexagons

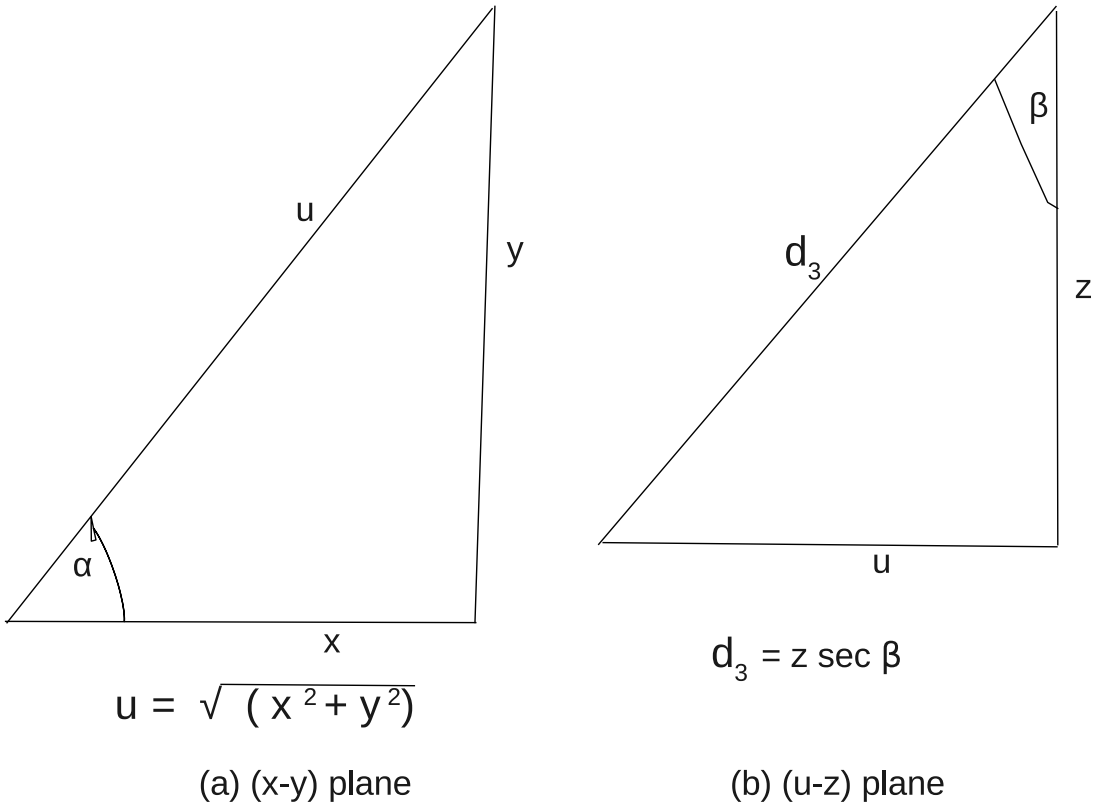


Figure 4.6 3D co-ordinates (x-y) plane and $(u - z)$ plane

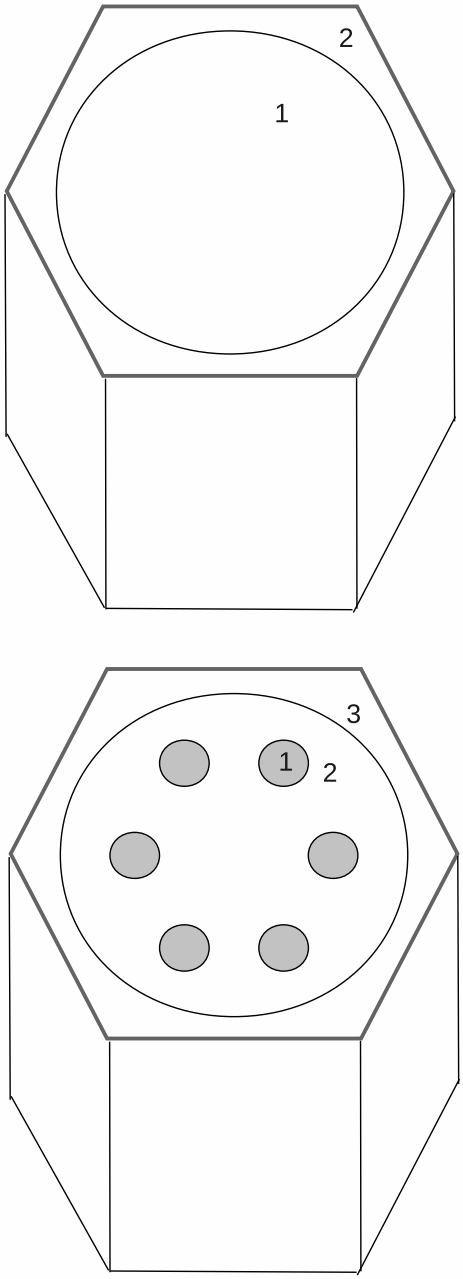


Figure 4.7 Numbering of Regions in 3D hexagon cells

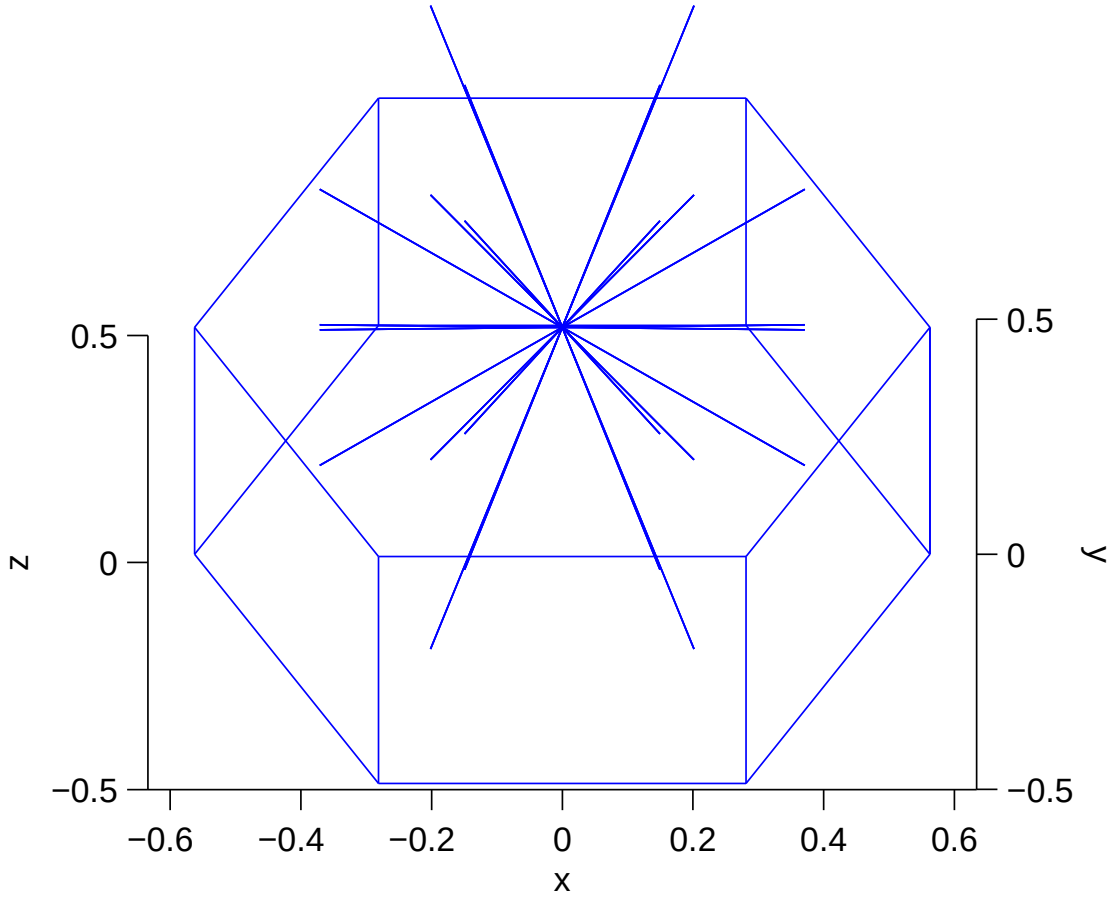


Figure 4.8 Tracks obtained inside a pin (side view)

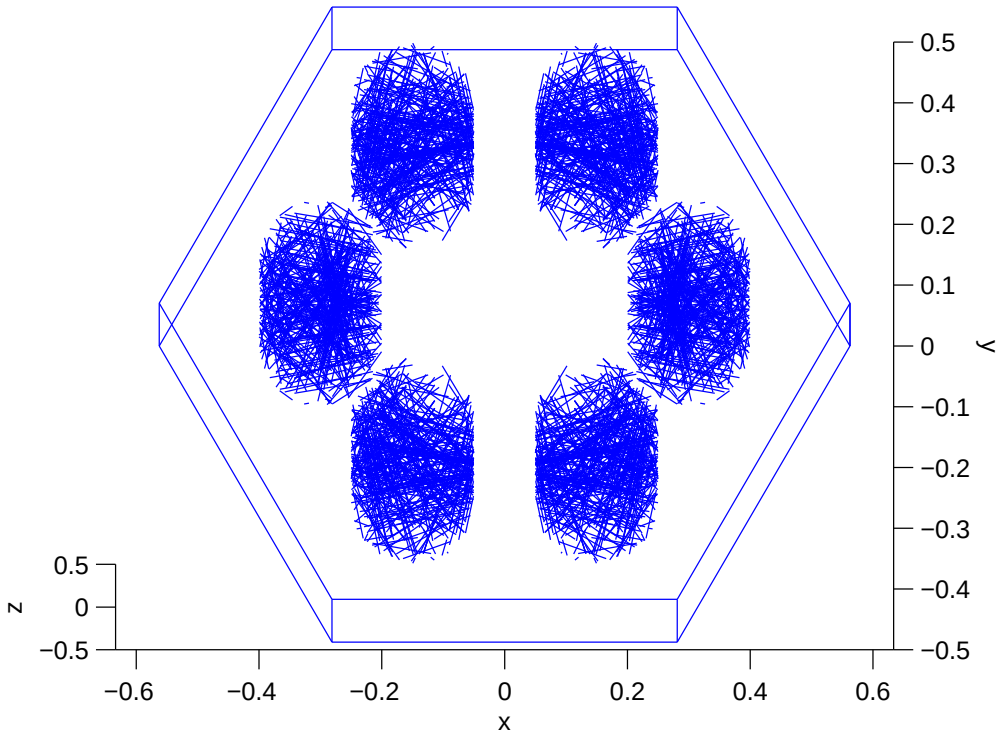


Figure 4.9 Tracks obtained in six pins in a 3D hexagon cell (side view)

CHAPTER 5

TRACKING ALGORITHMS AND COMPUTATION OF FLUXES IN SEVEN HEXAGON CELLS (2D AND 3D)

In this Chapter, we will describe the method of computing tracks in seven hexagons in two dimensions (2D) as well as in three dimensions (3D). Initially we compute tracks in each of the seven hexagons in two dimensions as described in Chapter 3. When we are required to compute tracks in 3D, then these 2D tracks are used to compute tracks in each of 3D hexagons described in chapter 4. The NXT module computes tracks in cylindrical pins described in chapter 2. Methodology and the results obtained by this approach will be presented in this chapter.

5.1 Methodology to compute tracks in Seven Hexagons

While using the method of collision probabilities, various lines are considered and track lengths in the hexagons are computed for each line. Before starting any computation, checking will be done to see if these hexagons are intersected by these lines.

5.1.1 Checking before computations:

First checking is done in 2D (x-y) plane. This is common for both the 2D and 3D cases because 2D cases are the projections of the line crossing 3D hexagons in the (x-y) plane. Checking is done to see if the tracking line intersects a circle which surrounds seven hexagons. Then it is checked, if the given line intersects circles which surround each of the seven hexagons individually as shown in Fig. 5.1. When the tracking line intersects with these circles, then only computations are done for finding the intersection points with the lines passing through the vertices of these seven hexagons. Equations for checking of the intersection points of a line with a circle are given in Appendix A.

5.1.2 Intersection points with seven hexagons

Computations in 2D for all the hexagons are done as mentioned in chapter 3. All these intersection points, in 2D plane are denoted as $((x_i, y_i), i = 1, 7)$. If it is a 3D case the co-ordinates are $((x_i, y_i, z_i), i = 1, 7)$. Here $((x_i, y_i), i = 1, 7)$ values are independent of z_i values. These $((x_i, y_i), i = 1, 7)$ values are first used to compute 2D track lengths. Then these 2D

track lengths and z_i values are used to compute track lengths in 3D case, for all the hexagons as mentioned in chapter 4.

For both the 2D and 3D computations, for multi hexagonal assemblies, the NXT module [23] of the code DRAGON [24] requires the information about the order in which the hexagons are encountered. For doing this we use all the x_i intersection points for all the hexagons. With each of these x_i values is associated its hexagon number i. Then all these x_i values we arrange either in decreasing or increasing order of their magnitude. In this way we come to know the order of the hexagon numbers encountered by the given line. For this purpose we use the bubble SORT routine [20]. All the x-intersection points are sorted by this routine. Here difference with the normal sorting is that it gives the number i associated with this x_i number. This way we come to know, in which order hexagons are intersected.

5.1.3 Computations for gap cases

In the case of multi hexagonal assemblies there may be some lines which go out of hexagons from one surface and re-enter other hexagons from outer surfaces as shown in Fig. 5.1. We call this case a gap case. In the case of seven hexagons, one gap is encountered for some lines, so there are two hexagons encountered by each gap. In the code DRAGON, it is assumed that this generates two independent lines crossing each of the two hexagons separately. Computations for both the hexagons are done separately. After computing such tracks, computation of tracks for other lines is done.

This process is repeated for all the lines and for all the hexagons, intersected by the given line. These tracks are used to compute collision probabilities which are used to compute fluxes in these hexagon cells as mentioned in chapter 2. All the computations are summarized in the flow chart Fig. 5.2.

5.2 Computations for seven (2D and 3D) hexagon cells

In this section, we present the results of one group calculations that have been made to check the accuracy of the algorithm, developed in this thesis, and implemented in the NXT module of the code DRAGON [24]. Cross-sections are taken from [33, 30]. These cross-sections are given for under moderated lattices and are presented in Table 5.1.

Each of seven hexagon cells consists of a cylindrical fuel pin of radius r surrounded by moderator region which fills the hexagonal cell (Figs. 5.3 and 5.4). Constant source is assumed in the moderated region.

a) Dimensions of the 2D hexagon cell are

- Side of the hexagon = d = 0.562563 cm.
- Radius of the cylindrical pin = r = 0.4 cm.

b) In the case of seven 3D hexagonal cells, dimensions selected are

- Side of the hexagon = d = 0.562563 cm.
- Height = 1.0 cm (-0.5 to +0.5)
- Radius of the cylindrical pin = r = 0.4 cm.

5.2.1 Plots of tracks obtained

Plots for 2D seven hexagon cell case are presented in Fig. 5.5. Plots for 3D case are presented in Figs. 5.6 and Fig. 5.7. In the Fig. 5.6 plots of tracks in one pin of seven 3D hexagon cells are presented. All these tracks are plotted in NXT module by the Matlab graphics. The selected track directions and track density per cm in this case are 3 and 10 respectively. In the case of 3D seven hexagonal cells the selected track directions and track density per cm are 8 and 8 (plots in central pin one central hexagon) and 2 and 8 (plots in central pin seven hexagon cells) respectively. These values are taken such that tracks are distinctively visible.

5.2.2 Comparison of results for seven hexagon cells (2D and 3D)

Average neutron fluxes are computed, for seven hexagon cells in 2D as well in 3D cases. In both the cases these are presented for 4 regions, since 12 regions have symmetric fluxes. Results have been obtained, for various values of N and T, described in chapter 2, by using both the modules EXCELT and NXT. These are presented for the case of seven 2D hexagon cells in Tables 5.2 and 5.3. For the case of seven 3D hexagon cells the results are presented in Tables 5.4 and 5.5. These results are compared in Table 5.6.

In the 3D case, reflective boundary conditions are applied at both the top and the bottom surfaces and average neutron fluxes are compared with 2D results.

The 2D and 3D EXCELT results are nearly same.

For 2D cases (Table 5.6), NXT results nearly match with the EXCELT results.

When NXT (2D) and NXT (3D) results are compared, it is obsereved that in the region 3, which is the outer cell fuel region, the results nearly match. For regions 1, 2 and 4, the 3D results are more than 2D results by 0.018, -0.017 and 0.012 respectively. The main reason is that NXT (3D) results are not converged.

This can be seen from Table 5.5, that in the case of results of NXT (3D), the error on surfaces appears to increase in these cases. The reason may be that in these cases some of the tracking lines are having α tending to $\pi/2$. The numerical computation of $\tan\alpha$ in 2D

case might not be computed properly in this case. When 3D tracks are computed by using these values, the error will get amplified, since 3D tracks are larger in value than 2D tracks.

Table 5.1 Cross-sections of fuel and moderator

Cross-section	Water	Fuel
Σ	1.0770816	0.677993
Σ_s	1.0767	0.44228

Table 5.2 Average fluxes (EXCELT) in 2D seven hexagon cells (each with one pin)

N	T	FLUX 1	FLUX 2	FLUX 3	FLUX 4
4	2.0	2.746	2.710	2.686	2.778
4	4.0	2.455	3.098	2.734	2.982
4	8.0	2.681	3.005	2.696	2.970
8	8.0	2.672	3.001	2.698	2.972
10	10.0	2.685	2.983	2.696	2.936
16	10.0	2.686	2.982	2.696	2.933
16	16.0	2.713	2.927	2.691	2.921
16	30.0	2.701	2.950	2.693	2.929
16	50.0	2.702	2.948	2.693	2.929

Table 5.3 Average fluxes (NXT) in 2D seven hexagon cells (each with one pin)

N	T	FLUX 1	FLUX 2	FLUX 3	FLUX 4	V	S
2	2.0	2.480	2.986	2.782	3.036	-0.2	0.5
4	2.0	2.561	3.010	2.698	3.118	-0.3	0.5
4	4.0	2.699	2.946	2.688	2.936	0.7	-2.6
4	8.0	2.725	2.938	2.692	2.954	0.03	0.3
8	8.0	2.716	2.929	2.687	2.917	0.06	0.3
10	10.0	2.712	2.934	2.690	2.925	0.03	0.2
16	10.0	2.712	2.934	2.691	2.926	0.03	0.1
16	16.0	2.706	2.941	2.693	2.926	0.01	0.1
16	30.0	2.694	2.959	2.693	2.930	0.02	0.03
16	50.0	2.700	2.950	2.693	2.929	-0.01	0.03

Table 5.4 Average fluxes (EXCELT) in 3D seven hexagon cells (each with one pin)

N	T	FLUX 1	FLUX 2	FLUX 3	FLUX 4
4	2.0	2.746	2.710	2.686	2.778
4	4.0	2.455	3.098	2.734	2.982
4	8.0	2.681	3.005	2.696	2.970
8	8.0	2.672	3.001	2.698	2.972
10	10.0	2.685	2.983	2.696	2.936
16	10.0	2.686	2.982	2.696	2.933
16	16.0	2.713	2.927	2.691	2.921
16	30.0	2.701	2.949	2.693	2.929
16	50.0	2.702	2.948	2.693	2.929

Table 5.5 Average fluxes (NXT) in 3D seven hexagon cells (each with one pin)

N	T	FLUX 1	FLUX 2	FLUX 3	FLUX 4	V	S
4	2.0	1.913	3.117	2.819	3.040	-2.7	-10.1
4	4.0	2.581	2.939	2.730	2.963	-0.5	-2.9
4	8.0	2.718	2.940	2.692	2.948	-1.1	-2.4
8	8.0	2.718	2.946	2.685	2.925	-0.7	-2.9
10	10.0	2.722	2.931	2.703	2.944	-0.4	-3.4
16	10.0	2.718	2.933	2.694	2.941	-0.1	-3.8

Table 5.6 Comparison of average fluxes in seven 2D and 3D hexagonal cells (each with one pin)

Module	FLUX 1	FLUX 2	FLUX 3	FLUX 4
EXCELT(2D)	2.702	2.948	2.693	2.929
EXCELT(3D)	2.702	2.948	2.693	2.929
NXT(2D)	2.700	2.950	2.693	2.929
NXT(3D)	2.718	2.933	2.694	2.941

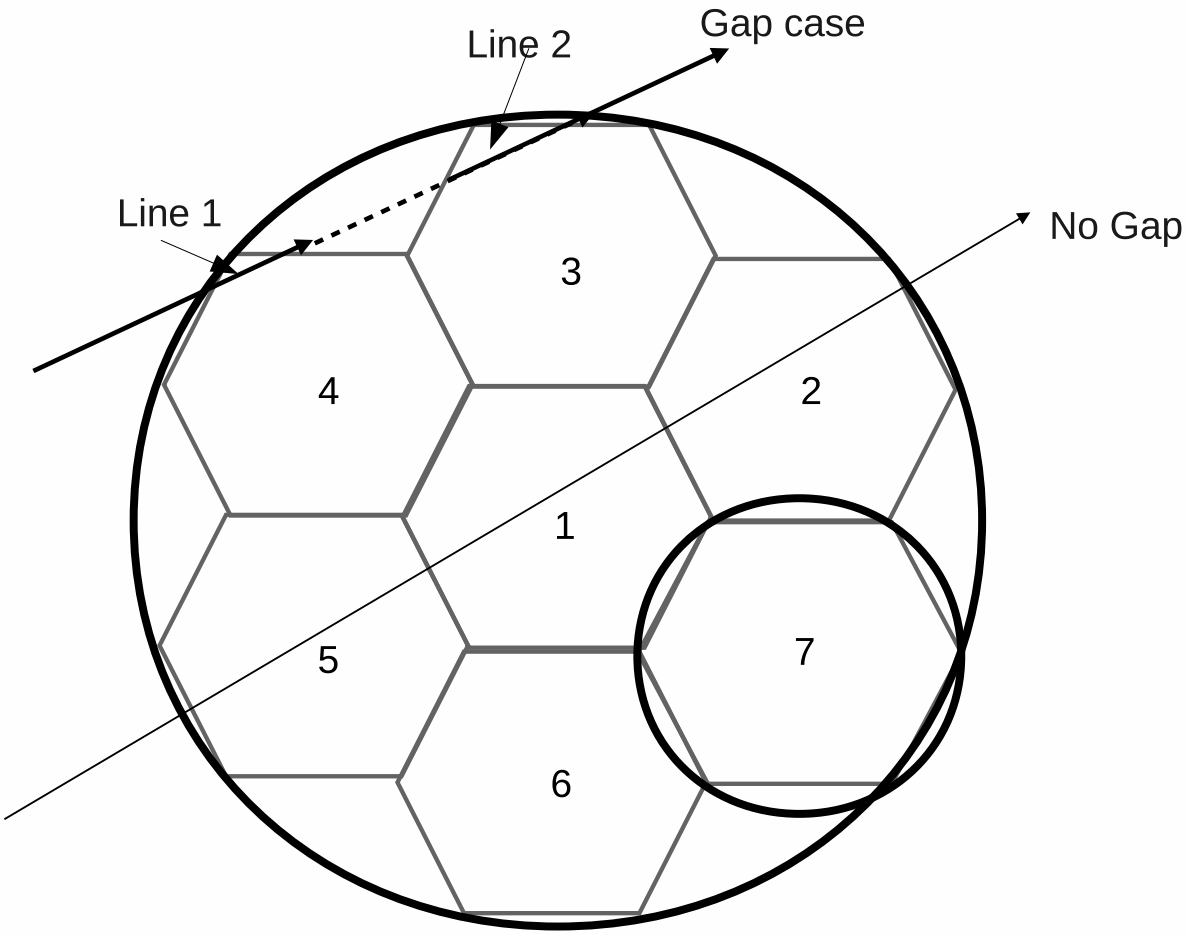


Figure 5.1 Circle surrounding seven hexagons

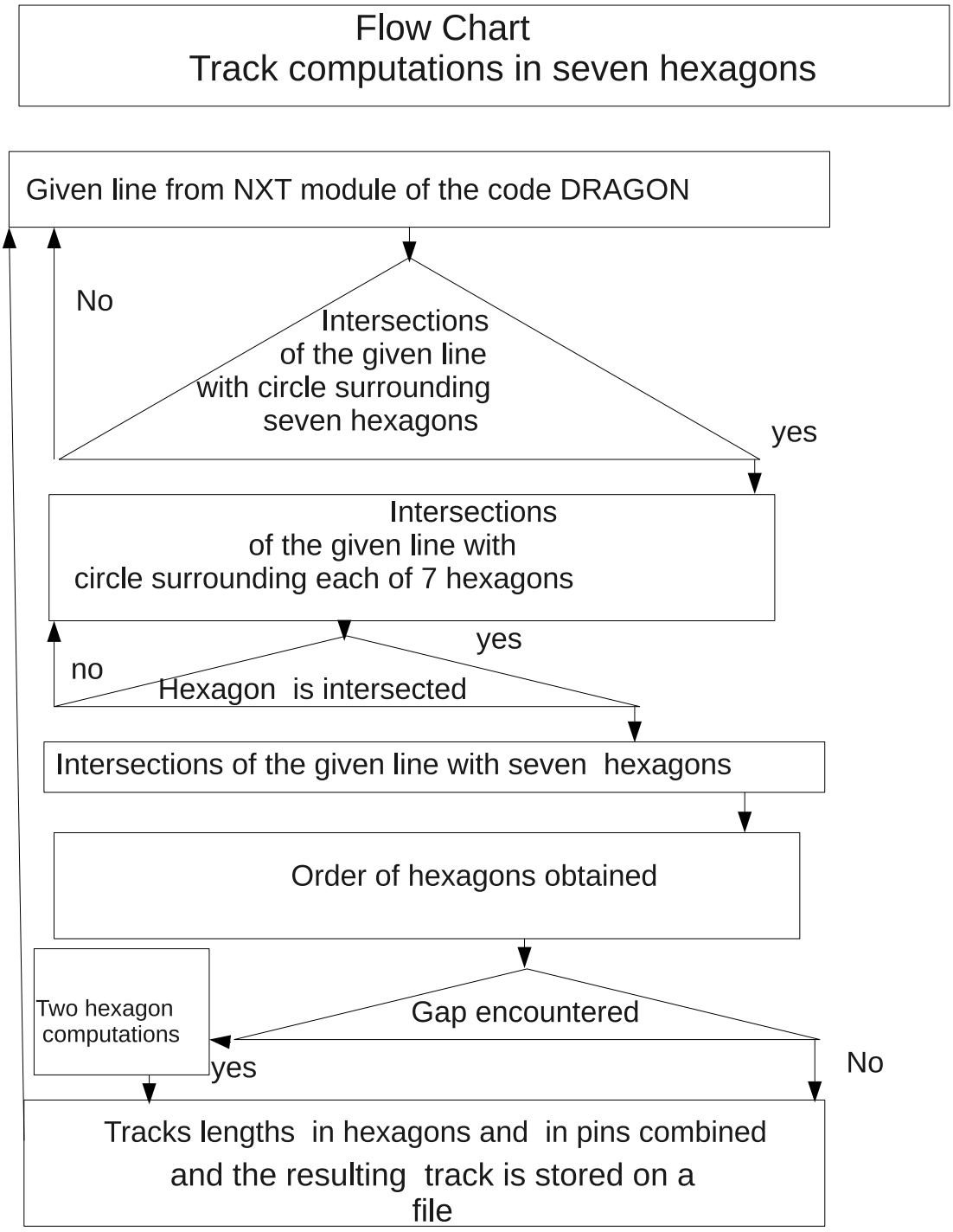


Figure 5.2 Flow chart for flux computations in seven hexagon cells

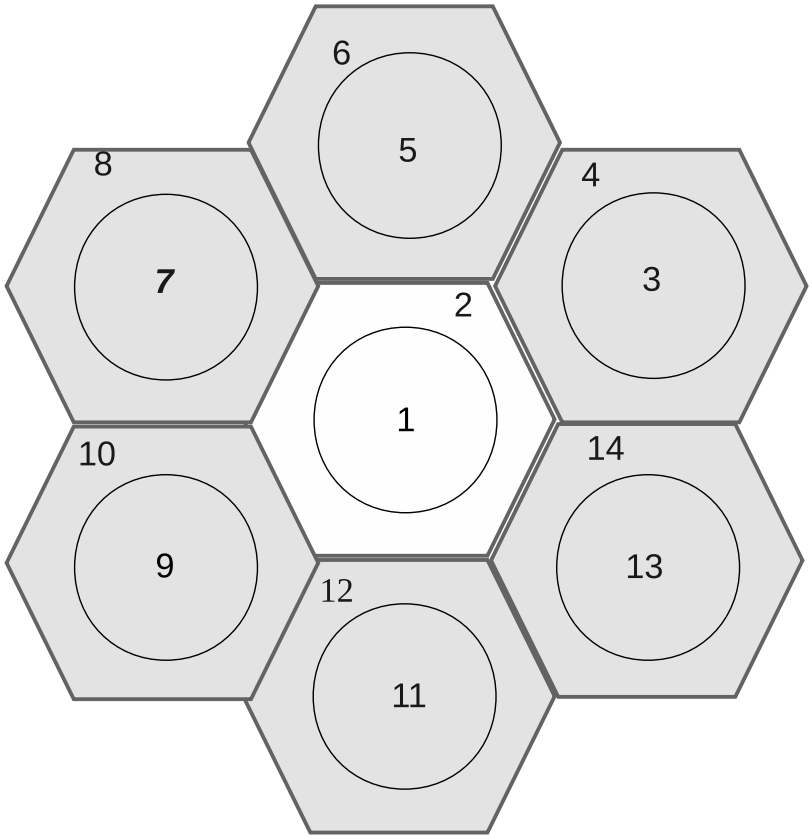


Figure 5.3 2D seven one pin hexagon cells

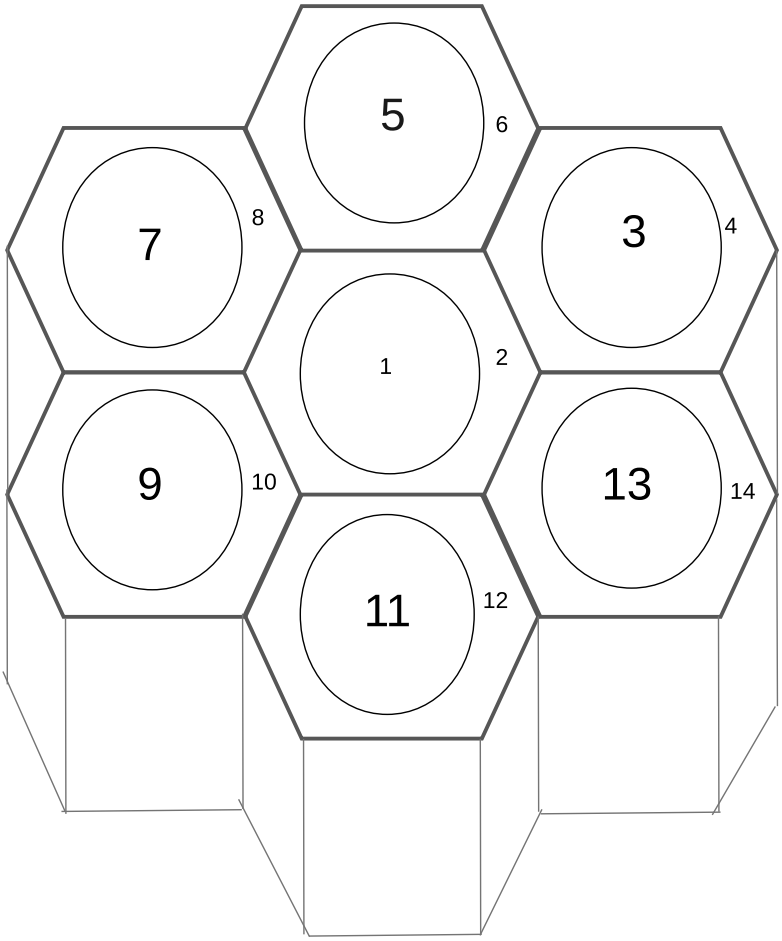


Figure 5.4 3D seven one pin hexagon cells

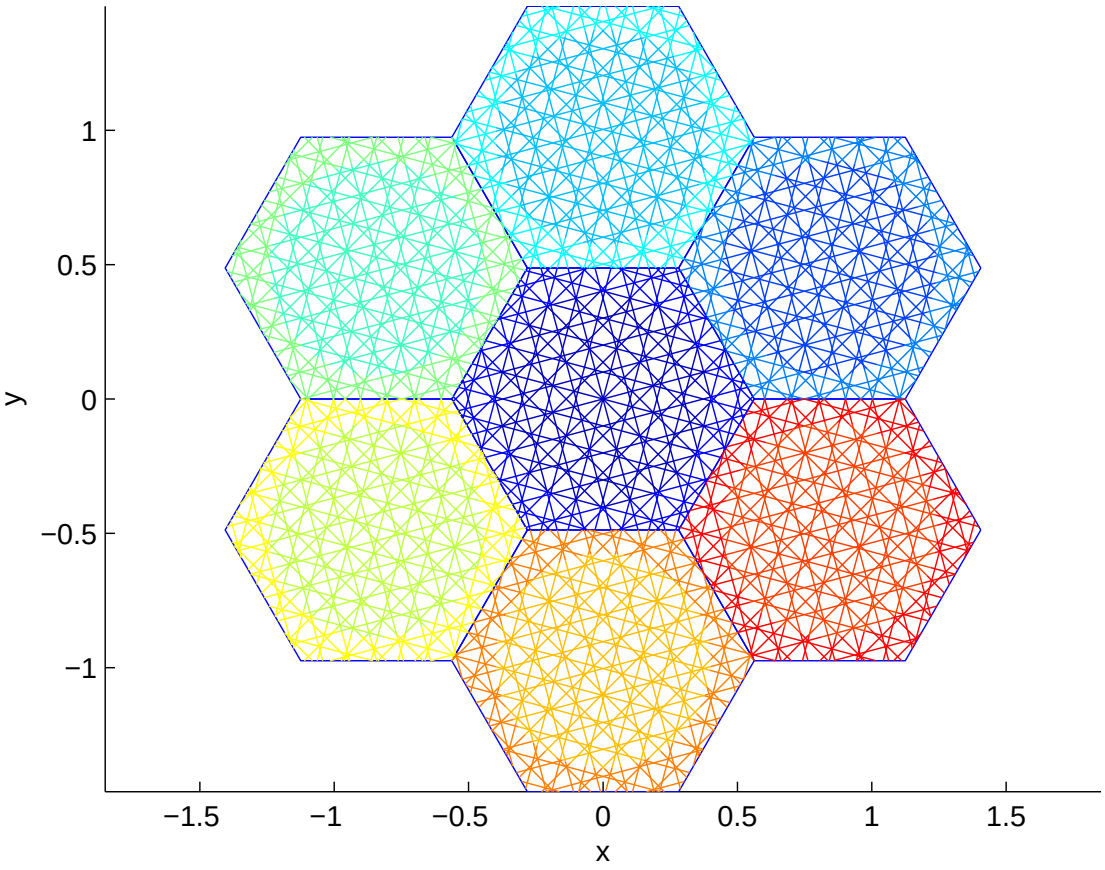


Figure 5.5 Tracks obtained in seven 2D one pin hexagon cells

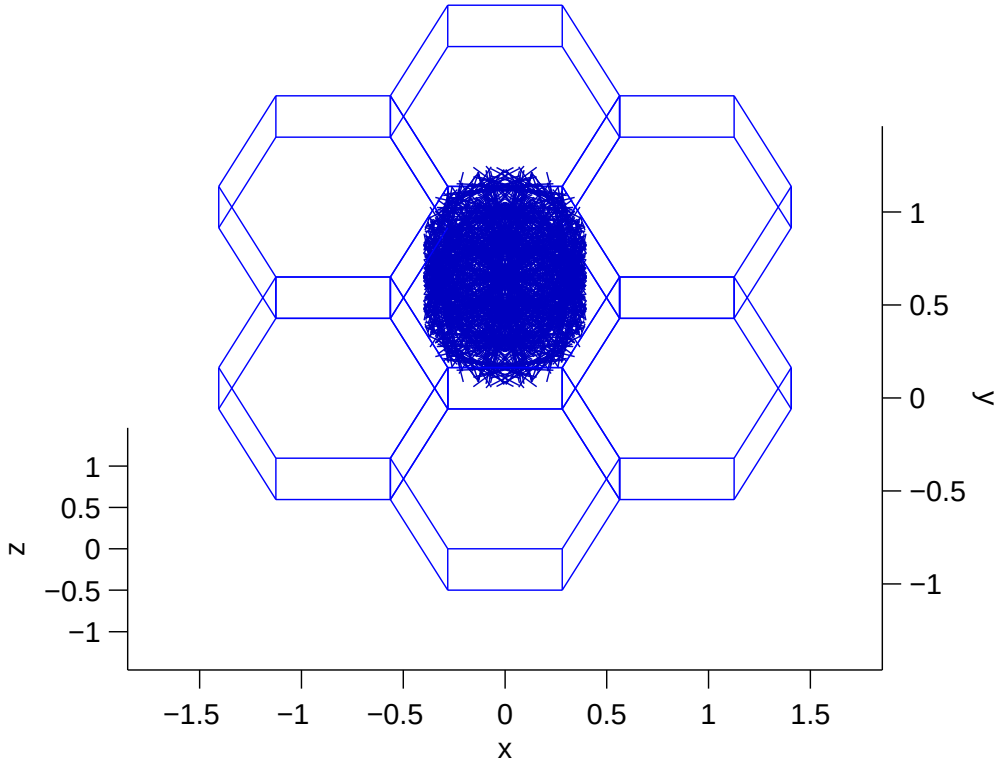


Figure 5.6 Tracks inside one pin of seven 3D one pin hexagon cells

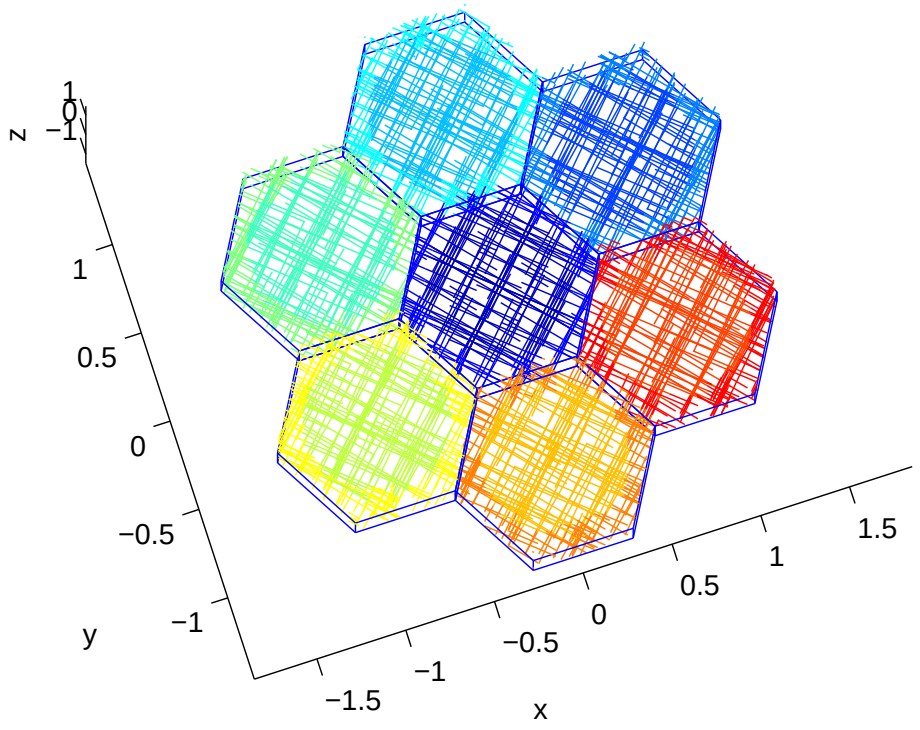


Figure 5.7 Tracks obtained in seven 3D one pin hexagon cells (Side view)

CHAPTER 6

CONCLUSIONS

The aim of this project was to develop tracking algorithms to compute tracks in multi hexagonal assemblies, in 2D and 3D, and implement them in the NXT module of the code DRAGON (Version 3.06).

Tracking algorithms have been developed and they have been implemented in the NXT module for seven hexagonal assemblies.

By using these algorithms, presented in the thesis, tracks lengths in multi hexagons are computed. The NXT module computes track lengths in one pin or a cluster of pins in a given geometry. Both of these track lengths are combined to compute tracks in multi hexagonal assemblies.

The co-ordinate systems for computing tracks in hexagons and in pins or cluster of pins, as the case may be, are different. Care has to be taken such that both the computations are done properly.

Our program is at a very preliminary stage. The data required for geometry specification is given manually and computations are done. This results in many pages of input preparation.

To understand all these things takes time, apart from the time required for the developement of tracking algorithms.

This methodology, developed and presented in the thesis has been programmed for seven hexagonal assemblies. This can be extended to compute tracks in any number of hexagon cells in 2D and 3D.

These algorithms could be extended to compute tracks in other geometries i.e. plate and Cartesian geometry (in 2D and 3D) etc.

REFERENCES

[1] ASKEW, J. (1972). A characteristics formulation of the neutron transport equation in complicated geometries. Technical report AAEW-M-1108, UKAEA, Winfrith.

[2] BICKLEY, W. and NAYLER, J. (1935). A short table of the functions $Ki_n(x)$, from $n = 1$ to $n = 16$. *Philos. Mag.*, 20, 343–347.

[3] BONALUMI, R. (1998). Integration of differential equations by the pseudo-linear (PL) approximation. *Ann. Nucl. Energy*, 25, 581–597.

[4] BRIESMEISTER, J. (2000). MCNP A general Monte Carlo N-Particle code, version 4c. Technical report LA-13709 M, Los Alamos Scientific Laboratory.

[5] CARLSON, B. and BELL, G. (1958). Solution of the transport equation by the S_n method. *Second International Conference on Peaceful Uses of Atomic Energy.*

[6] CARLSON, B. G. (1971). Tables of equal weight quadrature EQ_n over the unit sphere. Technical report LA-4734, Los Alamos Scientific Laboratory.

[7] CARLVIK, I. (1965). A method of calculating collision probabilities in general cylindrical geometry and application to flux distribution and Dancoff factors. *Third United Nation conference on peaceful uses of atomic energy, (Geneva).* vol. 2, 225.

[8] CHAN, P. S. W., TSANG, K. T. and BUSS, D. B. (2001). Reactor physics of NG CANDU. *22nd Annual CNS Conference.* (Proceedings available on CD-Rom).

[9] DAHMANI, M., LE TELLIER, R. and MARLEAU, G. (2008). Modeling reactivity devices for advanced CANDU reactors using the code DRAGON. *Ann. Nucl. Energy*, 35, 804–812.

[10] GIRARDINA, G., RIMPAULT, G., MORIN, F., BOSQ, J., CODDINGTON, P., MIKITYUK, K. and CHAWLA, R. (2008). Development and characterization of the control assembly system for the large 2400 MWth Generation IV gas-cooled fast reactor. *Ann. Nucl. Energy*, 35, 2206–2218.

[11] GLASSTONE, S. and ALEXANDER, S. (1981). *Nuclear reactor engineering.* Krieger Publishing Company.

[12] GUZMÁN, R. and FRANÇOIS, J.-L. (2007). Comparison between HELIOS calculations and a PWR cell benchmark for actinides transmutation. *Ann. Nucl. Energy*, 34, 22–27.

[13] HÉBERT, A. (2008). *Applied reactor physics*. Presses Internationales de l'École Polytechnique de Montréal.

[14] HOPWOOD, J. (2006). The advanced CANDU reactor ACR-1000. *PHYSOR 2006 ANS International Topical Meeting on Reactor Physics*. (Proceedings available on CD-Rom).

[15] HUDA, M. Q. and OBARA, T. (2008). Development and testing of analytical models for the pebble bed type HTRs. *Ann. Nucl. Energy*, 35, 1994–2005.

[16] JEHOUANI, A. and ELMORABITI, A. (2008). On the cylindrical symmetry of neutron fluxes in square and hexagonal reactor cells. *Ann. Nucl. Energy*, 35, 216–219.

[17] LE TELLIER, R. and HÉBERT, A. (2006). On the integration scheme along a trajectory for the characteristics method. *Ann. Nucl. Energy*, 33, 1260–1269.

[18] LEE, C. H., ZHONG., Z., TAIWO, T. A., YANG, W. S., SMITH, M. A. and PALMIOTTI, G. (2006). Status of reactor physics activities on cross-section generation and funtionalization for the prismatic very high temperature reactor, and development of spatially-heterogeneous codes. Technical report ANL-Gen IV-075, Argone National Laboratory.

[19] LI, H., HUANG, X. and LIANGJU, Z. (2008). A simplified mathematical dynamic model of the HTR-10 high temperature gas-cooled reactor with control system design purposes. *Ann. Nucl. Energy*, 35, 1642–1651.

[20] MAHAFFY, J. (1995). *Bubble sort routine*. `www.google.ca`.

[21] MARKL, H. (1965). New concepts in the application of collision probabilities in reactor theory. *Third International Conference on Peaceful Uses of Atomic Energy*.

[22] MARLEAU, G. (2001). DRAGON theory manual Part 1: Collision probability calculations. Technical report IGE-236 Rev. 1, École Polytechnique de Montréal.

[23] MARLEAU, G. (2005). New geometries processing in DRAGON the NXT module. Technical report IGE-260, École Polytechnique de Montréal.

[24] MARLEAU, G., ROY, R. and HÉBERT, A. (2008). A user guide for DRAGON 3.06D. Technical report IGE-174 Rev. 6D, École Polytechnique de Montréal.

[25] MARLEAU, G., ROY, R. and HÉBERT, A. (2009). A user guide for DRAGON Version 4. Technical report IGE-294, École Polytechnique de Montréal.

[26] MARTIN, N., MARLEAU, G. and HÉBERT, A. (2008). A preliminary study of the OECD/NEA 3D transport problem using the lattice code DRAGON. *2008 Symposium on Simulation Methods in Nuclear Engineering*. (Proceedings available on CD-Rom).

[27] MITTAG, S., PETKOV, P. T. and GRUNDMANN, U. (2003). Discontinuity factors for non-multiplying material in two-dimensional hexagonal reactor geometry. *Ann. Nucl. Energy*, 30, 1347–1364.

[28] NOH, J. M., KIM, K.-S., KIM, Y. and LEE, H. C. (2008). Development of a computer code system for the analysis of prism and pebble type VHTR cores. *Ann. Nucl. Energy*, 35, 1919–1928.

[29] OUISLOUMEN, M., MARLEAU, G., HÉBERT, A. and ROY, R. (1991). Computation of collision probabilities for mixed hexagonal-cylindrical geometries using the DP1 approximation to the $J_{\pm}$ technique. *International Topical Meeting on Advances in Mathematica, Computation and Reactor Physics*. 2.2.1.1–2.2.1.12.

[30] OUISLOUMEN, M., MARLEAU, G. and ROY, R. (1993). Applying the collision probability method in two and three dimensions. *Joint Intl. Conf. On Mathematical methods and supercomputing in Nucl. Appl. (Karlsruhe)*. vol. 1, 102–109.

[31] POPOV, E. L., YODER, G. L. and VELICHKOV, V. (2005). VVER-1000 three-dimensional RELAP5-3D model assessment. *Nuclear Technology*, 149, 304–308.

[32] ROY, R. (2000). The GAN generalized driver. Technical report IGE-158, École Polytechnique de Montréal.

[33] SANCHEZ, R. (1986). A transport multicell method for two-dimensional lattices of hexagonal cells. *Nucl. Sci. Eng.*, 92, 247.

[34] SCHULZ, T. (2006). Westinghouse AP1000 advanced passive plant. *Nucl. Sci. Eng.*, 2006, 1547–1557.

[35] SCHWINKENDORF, K. N. (2000). Calculation of the fast flux test facility fuel pin tests with the WIMS-E and MCNP codes. Technical report WHC-SA-1379-FP, To complete.

[36] SEKKI, D. and HÉBERT, A. (2007). A user guide for DONJON Version 4. Technical report IGE-300, École Polytechnique de Montréal.

[37] STAMM'LER, R. J. J. and ABBATE, M. (1983). *Methods of steady-state reactor physics in nuclear design*. Academic Press.

[38] TUCEK, K., CARLSSON, J. and WIDER, H. (2006). Comparison of sodium and lead-cooled fast reactors regarding reactor physics aspects, severe safety and economical issues. *Nuclear Engineering and Design*, 236, 1589–1598.

[39] VARIN, E., ROY, R. and KOCLAS, J. (2005). A user guide for DONJON Rev. 4. Technical report IGE-208, École Polytechnique de Montréal.

[40] VILLARINO, E. A., STAMM'LER, R. J., FERRI, A. A. and CASAL, J. J. (1992). HELIOS: Angularly dependent collision probabilities. *Nucl. Sci. Eng.*, 112, 16–31.

[41] VUJIC, J. L. (1991). Validation of GTRAN2 transport method for complex hexagonal assembly geometry. *Trans. Am. Nucl. Soc.*, 64, 523.

[42] WIMS (1999). A modular scheme for neutronics calculations. User's guide for version 8. Technical report ANSWER/WIMS(99)9, AEA technology.

APPENDIX A

CHECKING OF INTERSECTION POINTS OF A LINE WITH A CIRCLE

In this Appendix equations are developed to check if the given line intersects with a circle or not. This circle can pass through the vertices of one hexagon or it can surround the outer vertices of seven hexagons. If the line does not cut this circle, then the intersection of this line with the required hexagons will not be computed. The intersection points of the line with this circle are computed in the following way.

Let us consider a circle of radius r with its center at (a, b). Let us consider a line which makes an angle α ($M = \tan\alpha$) with x-axis and intercepts the y-axis at a distance of y_0.

Equation of a straight line is

$$y = Mx + y_0 \tag{A.1}$$

Equation of a circle is

$$(x - a)^2 + (y - b)^2 = r^2 \tag{A.2}$$

Let

$$A = (1 + M^2) \tag{A.3}$$

$$B = 2M(y_0 - b) - 2a \tag{A.4}$$

$$C = a^2 + (y_0 - b)^2 - r^2 \tag{A.5}$$

$$D = \sqrt{(B^2 - 4AC)} \tag{A.6}$$

Points of intersections are

$$x_i = (-B \pm D)/2A \tag{A.7}$$

$$y_i = Mx_i + y_0 \tag{A.8}$$

(where $i = 1, 2$)

If the parameter D ≤ 0 then the circle will not be intercepted by the given line. The given line will intersect the circle, only when D > 0.

APPENDIX B

EXPRESSIONS FOR COMPUTATION OF TRACKS IN 3D HEXAGONS

In this appendix, equations are given which are used in the computation of tracks in 3D hexagons. It is mentioned in chapter 4, that while computing tracks in a 3D hexagon, one is required to compute track distances in a rectangular geometry. We denote its four surfaces as top H_t, bottom H_b, L and R, as shown in Figs. B.1 and B.2. The z co-ordinates of the intersection points, on the two sides of the hexagon in $(z-u)$ plane, from the origin $(0,0,0)$ are given as z_1' and z_2'. Track distances are computed from the point (x_a, y_a, z_a). First track distance, of the given line from the point (x_a, y_a, z_a), is denoted as d_{31}. Second track distance, of the given line from this point is denoted as d_{32}. Difference of these two distances give us the required track length d_3, inside the 3D hexagon.

The direction of a line, in a 3D hexagon, will depend on the value of $\cos\alpha$ as mentioned in chapter 4. It can be seen from Fig. 4.2 that in 3D, two cases are considered: a) $\cos\alpha \geq 0$ and b) $\cos\alpha < 0$, depending on the direction of travel of the neutron.

The two cases are

a) $\underline{\boldsymbol{\cos\alpha \geq 0}}$

In this case a neutron travels either from the quadrant 5 to quadrant 1 or from quadrant 3 to quadrant 7. The boundaries encountered by the line for z_1' computations are either L or H_b, this can be seen from Fig. B.1. The position of z_1' is required to compute d_{31}. The boundaries encountered by the line for z_2' computations, required to compute d_{32}, are either R or H_t, it can be seen from this figure.

b) $\underline{\boldsymbol{\cos\alpha < 0}}$

In this case a neutron travels either from quadrant 7 to quadrant 3 or from quadrant 1 to quadrant 5. The boundaries encountered by the line for z_1' computations are either R or H_b, this can be seen from Fig. B.2. The position of z_1' is required to compute d_{31}. The boundaries encountered by the line for z_2' computations, required to compute d_{32}, are L or H_t.

The following equations are used to compute d_{31} and d_{32}.

a) Line cuts bottom and top surface respectively :

When $z_1' < H_b < z_2'$

$$d_{31} = (H_b - z_a)\sec\beta \tag{B.1}$$

When $z_1' < H_t < z_2'$

$$d_{32} = (H_t - z_a)\sec\beta \tag{B.2}$$

b) Line cuts left or right and top surface respectively:

When $H_t > z_1' > H_b$

$$d_{31} = (z_1' - z_a)\sec\beta \tag{B.3}$$

When $z_1' < H_t < z_2'$

$$d_{32} = (H_t - z_a)\sec\beta \tag{B.4}$$

c) Line cuts left and right surface respectively:

When $H_t > z_1' > H_b$

$$d_{31} = (z_1' - z_a)\sec\beta \tag{B.5}$$

When $H_t > z_2' > H_b$

$$d_{32} = (z_2' - z_a)\sec\beta \tag{B.6}$$

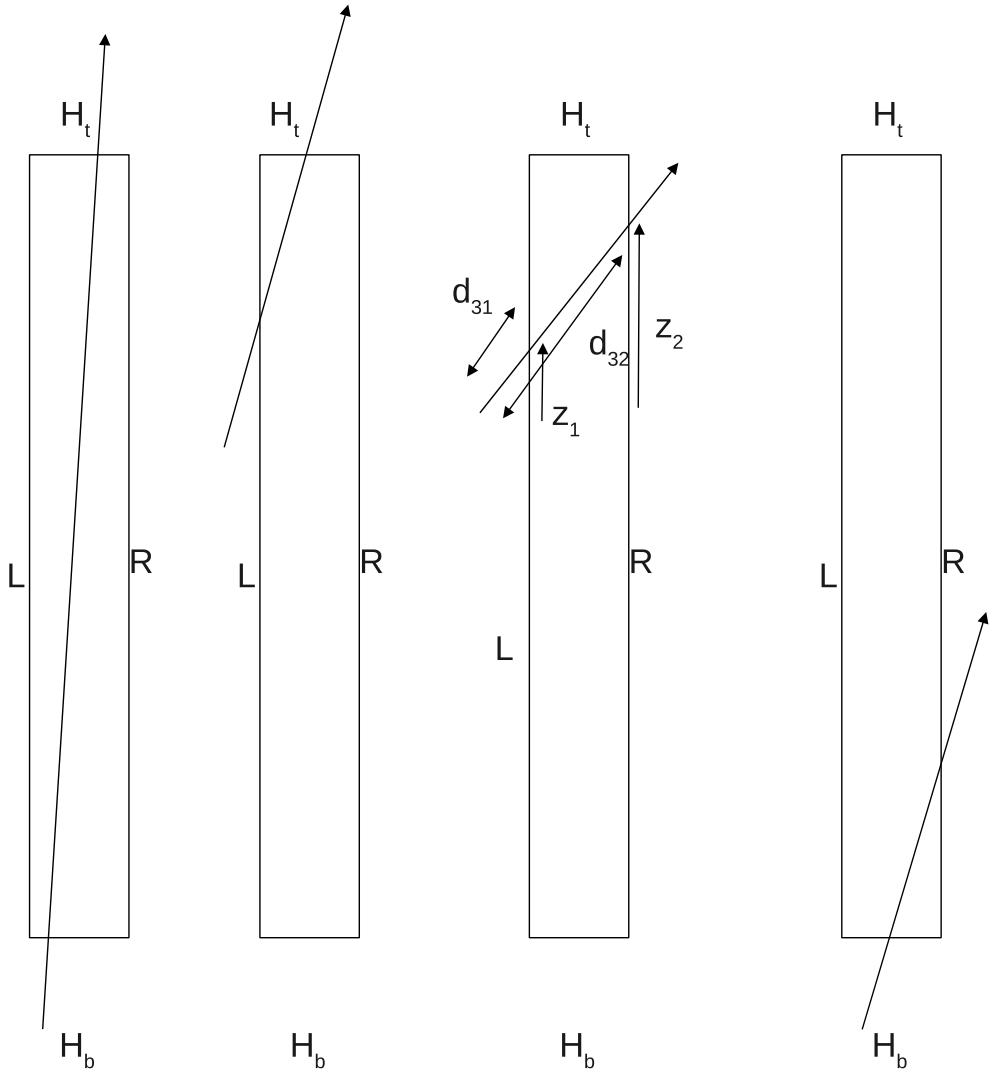
d) Line cuts bottom and right or left surface respectively:

When $z_2' > H_b > z_1'$

$$d_{31} = (H_b - z_a)\sec\beta \tag{B.7}$$

When $H_t > z_2' > H_b$

$$d_{32} = (z_2' - z_a)\sec\beta \tag{B.8}$$



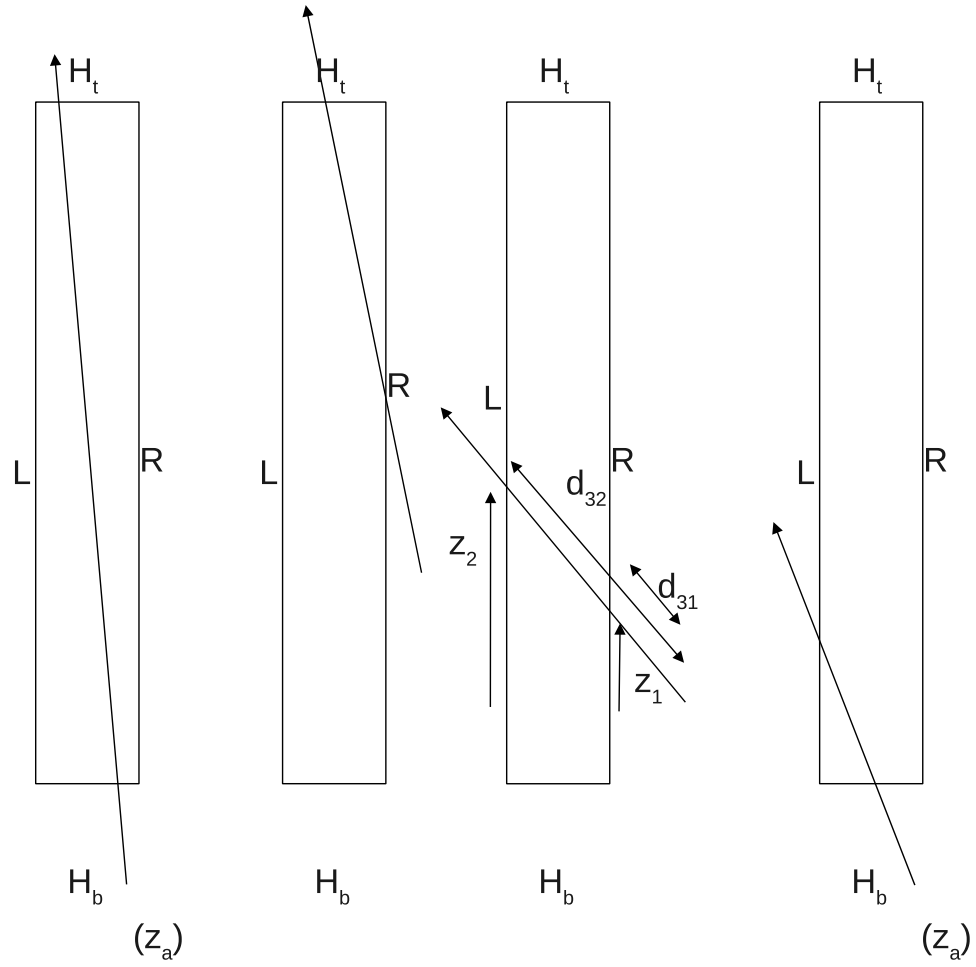
(a) Surfaces H_b and H_t

(b) Surfaces L and H_t

(c) Surfaces L and R

(d) Surfaces H_b and R

Figure B.1 Intersection of lines with rectangular boundaries for $\cos\alpha \geq 0$



(e) Surfaces H_b and H_t

(f) Surfaces H_t and R

(g) Surfaces R and L

(h) Surfaces H_b and L

Figure B.2 Intersection of lines with rectangular boundaries for $\cos\alpha < 0$